Sonderheft Nr. 247
Preis DM 19.50
öS 158.–, sfr. 19.50
mc

DAS EMUF-SONDERHEFT 2

EINPLATINEN-MIKROCOMPUTER FÜR UNIVERSELLE FESTPROGRAMM-ANWENDUNG

6504-, 6502-, Z80-, 68008-, 8086- und Basic-EMUF

Telefon-EMUF

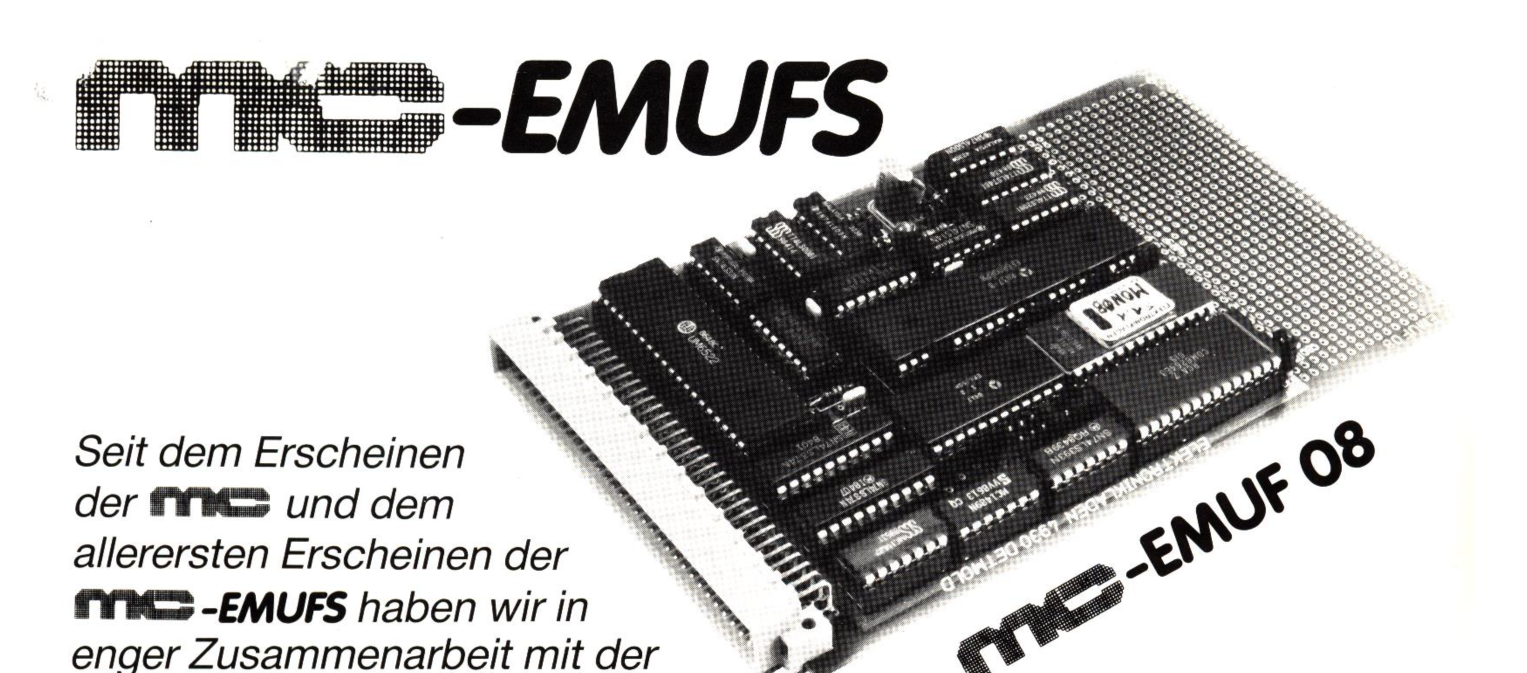
Universelle Fernbedienung

Basic-EMUF mit LCD und Tasten

EMUF als Bordcomputer

Z80-EMUF mit Display und Tastatur

Multitasking mit dem Z80-EMUF



Seit dem Erscheinen der mc und dem allerersten Erscheinen der mc ***-EMUFS*** *haben wir in enger Zusammenarbeit mit der Redaktion an der Entwicklung dieser Einplatinencomputer gearbeitet. Wenn Sie sich also für diese günstigen Einplatinencomputer interessieren, dann sollten Sie sich mit uns in Verbindung setzen. Rufen Sie unsere kostenlose Broschüre* ***„VON EMUFS & EPACS“*** *ab. Wir bieten Ihnen u. a.*

EMUF 6504

Bausatz	89,– DM
Fertigkarte	119,– DM
Leerplatine	30,– DM

EMUF 86

Bausatz	269,– DM
Fertigkarte	369,– DM
Leerplatine	100,– DM

EMUF 08

Bausatz	159,– DM
Fertigkarte	248,– DM
Leerkarte	48,– DM

BASIC-EMUF

Bausatz	ab 198,– DM
Fertigkarte	ab 390,– DM
Leerkarte + GAL	98,– DM

EMUF 6502/232

Bausatz	ab 109,– DM
Leerplatine	39,– DM

EMUF 6502

Bausatz	99,95 DM
Leerplatine	39,– DM

Andere Einplatinenrechner finden Sie in unserer Broschüre. Folgende CPUS stehen zur Auswahl: 6502, 6504, Z80A, NSC 800, HD 64180, 6805, 6809, 8052, 8086 (V30), 68008 und 68000.

oder von unseren Verkaufsstellen:

1000 Berlin 21, Rostocker Straße 31,
Telefon 030/3923011

4400 Münster, Hammer Straße 157,
Telefon 0251/795125

8000 München 19, Schulstraße 28
Telefon 089/1679499



Vorwort

Einem Computer-Freak den Begriff EMUF zu erklären hieße Eulen nach Athen tragen. In den sechs Jahren, seit der „Einplatinen-Computer für universelle Festprogramm-Anwendungen" aus der Taufe gehoben wurde, sind allein von der 6504-Urversion an die 10 000 Stück verkauft worden. Mittlerweile gibt es eine ganze EMUF-Familie mit so unterschiedlichen CPUs wie 6502, Z80, 68008 und 8086. Das Erfolgsrezept ist bei jedem Produkt das gleiche.

Erstens: Der Preis stimmt. Komplette Bausätze gibt es nach wie vor für weniger als 100 DM.

Zweitens: Für die Entwicklung von Programmen eignen sich gängige Tischcomputer. In der Anfangszeit waren das Systeme wie der AIM-65 und der Apple – jetzt ist zum Beispiel der IBM-PC hinzugekommen.

Drittens: Für viele der früher und in diesem Sonderheft beschriebenen Anwendungen gibt es fertige EPROMs. Man muß also nicht unbedingt programmieren können, um einen EMUF sinnvoll einsetzen zu können.

Dieses Sonderheft faßt sämtliche Grundlagenbeiträge sowie die interessantesten und aktuellsten Anwendungen aus der mc zusammen. Darüber hinaus stellt es zwei neue EMUFs vor: den EMUF86 und den EMUF232.

Zusammen mit dem Basic-EMUF ist mit dem EMUF86 wohl ein vorläufiger Höhepunkt erreicht. Dieses System stellt einen IBM-PC von der Leistungsfähigkeit her glatt in den Schatten. Aber auch die „alten Platinchen" sind nach wie vor aktuell. Man muß ja nicht gleich einen Ziegelstein mit dem Lastwagen transportieren.

Ihre Redaktion

Inhalt

Impressum: 1. Auflage 1987, Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2.
Bearbeitet von der Redaktion der Zeitschrift mc. Für den Text verantwortlich: Dipl.-Ing. (FH) Rudolf Hofer.

ISSN 0722-0022. ZV-Art.-Nr. 24704. Druck-Nr. F/ZV/987/1287/8'

Herwig Feichtinger

Von der Idee zum Programm

Wie entsteht eine typische Applikation für einen Einplatinen-Computer? An einem einfachen Beispiel, einer Steuerung für eine Relaisfunkstelle, wollen wir uns das einmal ansehen – inklusive aller typischen Schwierigkeiten, die dabei auftreten.

Die Aufgabenstellung lautet: Eine Relaisfunkstelle ist mit einer möglichst preiswerten und einfach zu bauenden Steuerung zu versehen, die die Aufgabe hat, auf ein Empfangs-Signal zu warten, den Sender bei Bedarf einzuschalten, automatisch das postalisch geforderte Relaisfunkstellen-Rufzeichen im Morsecode auszustrahlen und auf Abruf (nach Empfang eines Tonrufes) einen vorher fest programmierten Text, z. B. mit technischen Daten der Funkstelle, ebenfalls im Morsecode zu senden. Wird kein Signal mehr empfangen, so muß der Sender mit einer Verzögerung (Haltezeit) von etwa drei Sekunden wieder ausgeschaltet werden. *Bild 1* zeigt das Blockschaltbild des Systems.

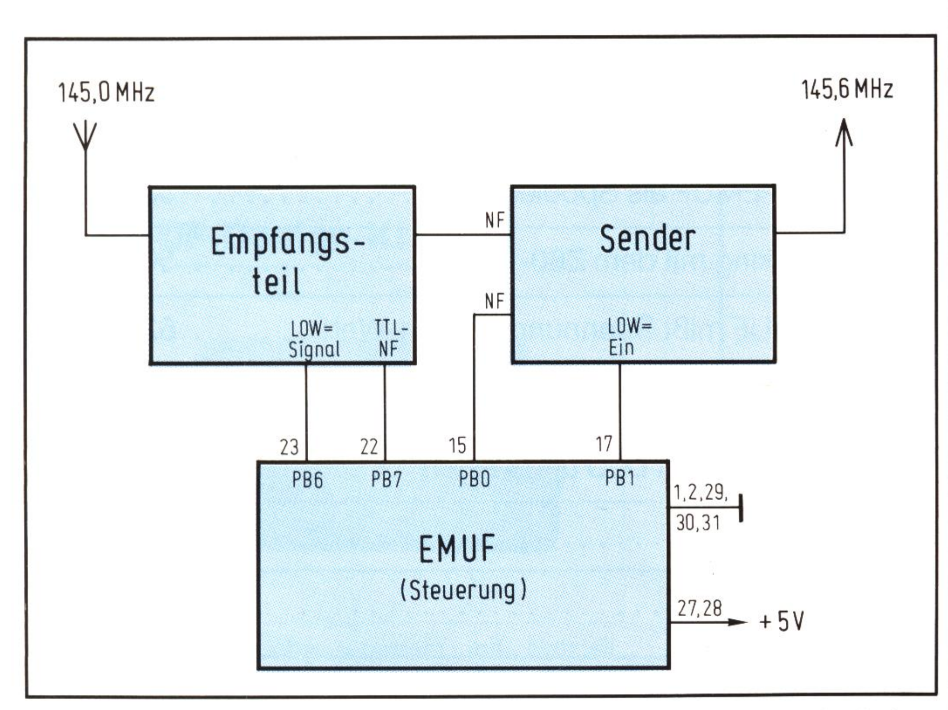


Bild 1. Blockschaltbild einer mit dem Einplatinen-Computer EMUF realisierten Relaisfunkstelle. Der EMUF enthält ein Rauschsperren-Signal vom Empfänger, das ihm sagt, ob ein Signal empfangen wird, sowie die Niederfrequenz, um einen 1750-Hz-Tonruf decodieren zu können. An den Sender liefert er ein Ein-/Aus-Signal sowie die Modulations-Nf der erzeugten Morsezeichen

Welche Funktionen sind per Software realisierbar?

Es gibt in vielen Anwendungen die Möglichkeit, bestimmte Teilaufgaben entweder per Software vom Mikrocomputer oder per Hardware von speziellen Bauelementen übernehmen zu lassen. Bei unserer Relaisfunkstelle ist es naheliegend, rein digitale Schaltfunktionen (Sender ein/aus) vom Mikrocomputer ausführen zu lassen. Es müssen aber auch Töne erzeugt (Morsezeichen-Aussendung) und erkannt werden (1750-Hz-Tonruf zum Einschalten des Senders und zum Abruf des Festtextes). Diese beiden Aufgaben könnten mit Hardware-Oszillator (z. B. NE555) und Phase-Locked-Loop-Tondecoder (z. B. NE567) ausgeführt werden, wenn dem Programmierer dazu keine Software-Lösung einfällt oder der Prozessor dafür zu langsam wäre.

Eine Alternative ist die Verwendung von Software zur Tonerzeugung (kein Problem mit Programmschleifen) und zur Tondecodierung (hier gelöst mit dem Verfahren der Autokorrelation, vgl. [1]). Daß das entwickelte Programm auf Anhieb das tut, was man sich ursprünglich vorgestellt hat, ist sehr unwahrscheinlich und bei längeren Programmen praktisch auszuschließen. Typische Fehlermöglichkeiten sind:

a) Fehler bei der Erstellung des Ablaufplans. Es kann sein, daß es sich herausstellt, daß es in der Praxis eben doch nicht so geht, wie man es sich an Hand des Flußdiagramms vorgestellt hat, z. B. wegen Zeitproblemen oder logischen Irrtümern.
b) Fehler bei der Erstellung des Assemblerprogrammes aus dem Flußdiagramm. Möglicherweise hat man das Flußdiagramm doch zu grob gezeichnet und so die Übersicht verloren. Eine kleine Hilfe: Zeichnen Sie nachträglich verwendete Assembler-Symbole (Labels) an die entsprechenden Stellen des Flußdiagramms ein.
c) Syntaktische Fehler. Diese sind am leichtesten auszuräumen, weil sich der Assembler während der Übersetzung des Quellencodes in den Objektcode automatisch darüber beschwert.

Der Hardware wird bei vielen Lösungen der Vorzug gegeben, soweit es sich nur um Einzelstücke handelt, da die Kosten der zusätzlich benötigten Bauelemente geringer sind als die zeitintensive Erarbeitung von Software-Know-How über Spezialaufgaben durch den Entwickler. Bei größeren Stückzahlen treten dagegen

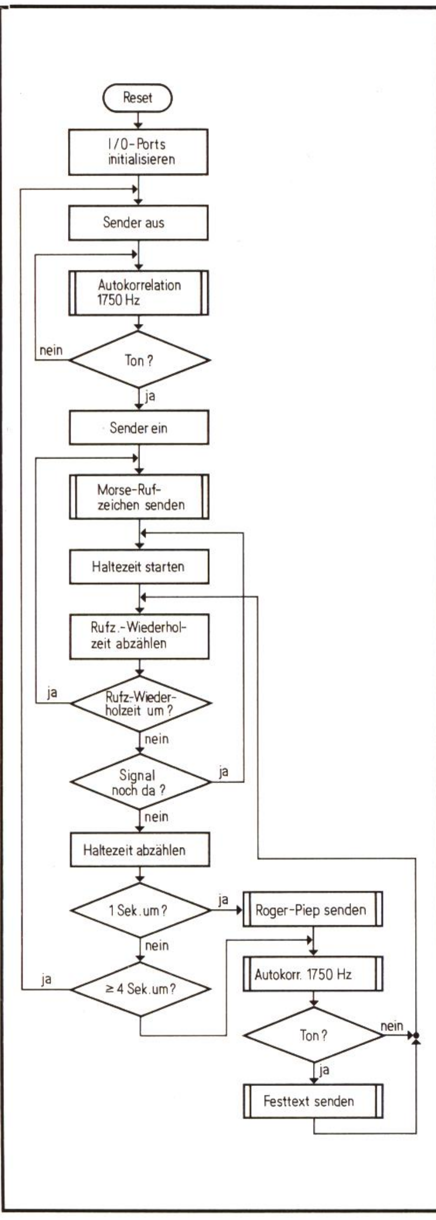


Bild 2. Flußdiagramm der EMUF-Software zur Steuerung der Relaisfunkstelle. Für die Autokorrelation und die Morsezeichen-Tonerzeugung werden Unterprogramm-Module verwendet

die Bauelemente-Kosten in den Vordergrund gegenüber den nur einmal auftretenden Entwicklungskosten.
In unserem Fall der Relaisfunkstelle wurden praktisch alle Teilaufgaben so weit wie möglich per Software realisiert, so daß der Hardware-Aufwand minimiert ist.

Zweiter Schritt: Programmablaufplan

Hat man sich überlegt, welche Teilaufgaben des Gesamtprojekts überhaupt vom Mikrocomputer übernommen werden sollen, kann man daran gehen, einen groben Programmablaufplan in Form eines Flußdiagramms oder Struktogramms zu erstellen. Dies sollte man auch bei kleineren Problemen unbedingt tun, um später nie die Übersicht zu verlieren. *Bild 2* zeigt den groben Ablauf der Software. Wie detailliert die Darstellung erfolgt, hängt von der Art der Problemstellung, dem Abstraktionsvermögen des Programmierers und der Art der bereits vorentwickelten und vielleicht irgendwo schon veröffentlichten Programm-Module, etwa zur Autokorrelation, ab. Sollten sich später bei Praxistests Änderungen im Ablauf ergeben, sollte man diese unbedingt auch im Flußdiagramm vermerken, um die Dokumentation auf dem tatsächlichen Stand der Dinge zu halten.

Programmerstellung mit Entwicklungssystem

Normalerweise steht bis zu diesem Zeitpunkt auch fest, welchen Computer man zur Lösung des Problems einsetzen möchte; in unserem Beispiel fand ein mc-EMUF Verwendung [2]; ein „Einplatinen-Mikrocomputer für universelle Festprogrammierung", der die mit dem bekannten 6502 voll software-kompatible CPU 6504 verwendet und knapp 100 DM kostet. Da der Einplatinencomputer selbst nicht dafür ausgelegt ist, Programme zu entwickeln, sondern sein Betriebsprogramm lediglich als EPROM erhält, und außerdem nicht für höhere Programmiersprachen wie Basic oder Pascal ausgelegt ist, weil dafür der Speicherplatz nicht ausreichen würde (außerdem wäre dann die Verarbeitungsgeschwindigkeit zu gering), muß man einen Tischcomputer verwenden, um das nötige Maschinenprogramm zu entwikkeln und in ein EPROM zu brennen.
Besitzt das Entwicklungssystem die gleiche CPU wie der Einplatinencomputer oder eine software-kompatible Version, so kann es auch während der Entwicklungsphase den Einplatinencomputer ersetzen, d. h. vorübergehend über seine I/O-Anschlüsse die Relaisfunkstelle selbst steuern. Stellt sich heraus, daß noch Änderungen am Programm vorzunehmen sind, so braucht man dann nicht jedesmal das EPROM zu löschen und neu zu programmieren.
Für die Entwicklung von Maschinensprache-Programmen von mehr als etwa 100 Byte Länge ist die Methode, Befehlscodes in Programmierhandbüchern einzeln nachzusehen und nur hexadezimale Bytes in den Computer zu hacken, in keiner Weise effektiv. Vielmehr sollte man sich eines Assemblers bedienen, der die Übersetzung mnemonischer Befehle (LDA, STA, JMP usw.) in hexadezimale Bytes (A5, 8D, 4C, usw.) automatisch vornimmt. Er ermöglicht auch das spätere Einfügen neuer Befehle, ohne umständlich alle Adressen neu berechnen zu müssen.
In unserem Beispiel, der mit dem EMUF realisierten Relaisfunkstellen-Steuerung, diente ein AIM-65 von Rockwell zusammen mit einem Kassettenrecorder und einem EPROM-Programmierzusatz als preiswertes Entwicklungssystem (Gesamtkosten ca. 1400 DM). Das komplette Programm ist in [3] abgedruckt.

Test des Systems

Hat man das Programm auf dem Entwicklungssystem feriggestellt und vom Assembler übersetzt, so kann man es, wie schon erwähnt, noch mit ihm testen, indem man die I/O-Ports des Entwicklungssystems als Ersatz für diejenigen des Einplatinencomputers anschließt. Läuft alles soweit zur Zufriedenheit, so muß man im Assembler-Quellencode die Adressen der Entwicklungssystem-I/O-Ports und eventuell verwendete Timer-Adressen auf die entsprechende Belegung des Einplatinen-Computers ersetzen, das Quellenprogramm vom Assembler nochmals in Objektcode übersetzen lassen (der nun so nicht mehr auf dem Entwicklungssystem, sondern nur noch auf dem Einplatinencomputer ablauffähig ist) und ein EPROM damit programmieren. Beim Einschalten des nun mit dem Einplatinencomputer verbundenen Systems dürften dann keine Probleme mehr auftreten, sofern es sich um ein erprobtes Einplatinen-System wie den EMUF handelt. Andernfalls kann man beliebig viel Geld für Fehlersuch-Einrichtungen ausgeben – vom Mehrkanal-Oszilloskop über In-Circuit-Emulatoren bis zu Logik-Analysatoren. Die Verwendung fertiger, bereits erprobter Einplatinen-Computer kann somit einige zehntausend DM sparen.

Literatur

[1] Tonerkennung per Software (Autokorrelation). mc 1981, Heft 4.
[2] Mädchen für alles (EMUF). mc 1981, Heft 2, oder EMUF-Sonderheft.
[3] EMUF steuert Relaisfunkstelle. EMUF-Sonderheft, Franzis-Verlag.
[4] AIM schießt EPROM. EMUF-Sonderheft, Franzis-Verlag.

Herwig Feichtinger

Mädchen für alles

Was hier im folgenden vorgestellt wird, ist ein fest zu programmierender, sehr preiswerter Mikrocomputer, der sich zum Beispiel als Drucker-Interface, intelligentes Bedienteil für Meßgeräte, Frequenzgenerator, Schaltuhr, Codeumsetzer oder für tausend andere Zwecke einsetzen läßt. Die Programme für ihn lassen sich mit preiswerten Tischcomputern auf 6502-Basis wie Apple, PET, CBM, AIM-65, PC-100 oder KIM-1 entwickeln.

Wenn man von Computern spricht, meint man meist Geräte, die sich frei programmieren lassen, mit denen man eigene Programme entwickeln und testen kann und die über eine Tastatur sowie über einen Bildschirm oder wenigstens ein einfaches Display verfügen. Solche Computer bekommt man heute schon für weniger als 1000 DM.
Hier wird aber etwas ganz anderes vorgestellt, nämlich ein Mikrocomputer, der nur einmal und vor allem fest programmiert und dann für einen ganz bestimmten Verwendungszweck eingesetzt wird (*Bild 1*). Er ist also in keiner Weise dafür konstruiert, Programme mit ihm zu entwickeln, als Lehr- und Lerncomputer zu dienen oder später mit zusätzlichem Speicherplatz, ja vielleicht sogar mit einem Basic-Interpreter erweitert zu werden.

Bild 1. Gesamtschaltung des 6504-Computers mit 1 KByte ROM, 128 Byte RAM, einem programmierbaren Interrupt-Timer und 16 I/O-Leitungen

Ein Computer für weniger als hundert Mark

Unser Computerchen ist also dafür gedacht, überall dort eingesetzt zu werden, wo es im Grunde nur als Ersatz für eine vielleicht recht umfangreiche, undurchsichtige Digitalschaltung dient. So etwa in einer numerischen Steuerung, in einer Schaltuhr, in einem rechnenden Meßgerät usw., wo der Benutzer nicht selbst programmiert.
Dieses Konzept gestattet es, einen Mikrocomputer als Minimalkonfiguration mit absichtlichem Verzicht auf spätere Erweiterbarkeit und gleichzeitig als äußerst preiswerte Schaltung aufzubauen. Natürlich gibt es für diesen Zweck auch Ein-Chip-Mikrocomputer, z. T. sogar mit UV-löschbaren EPROMs – aber: ein Entwicklungssystem für einen solchen Computer kostet leider -zigtausend Mark. Bei geringen Stückzahlen treten daher enorme Kostenbelastungen auf, die die Verwendung der Ein-Chip-Mikrocomputer wieder oft als fraglich erscheinen lassen.
Unser Mikrocomputer arbeitet daher mit einer CPU, die es zuläßt, die benötigten Programme mit preiswerten Tischcomputern zu entwickeln, so etwa mit CBM, PET, AIM-65, Apple-II usw., die alle mit dem Mikroprozessor 6502 arbeiten. Bei der Übertragung des Programms auf das EPROM, das in unser Computerchen gesteckt wird, brauchen dann lediglich noch einige Adressen geändert zu werden. Zum Beispiel diejenigen für die I/O-Ports. Verwendet man einen Assembler für die Programmentwicklung, so braucht man das nicht einmal einzeln von Hand zu tun.

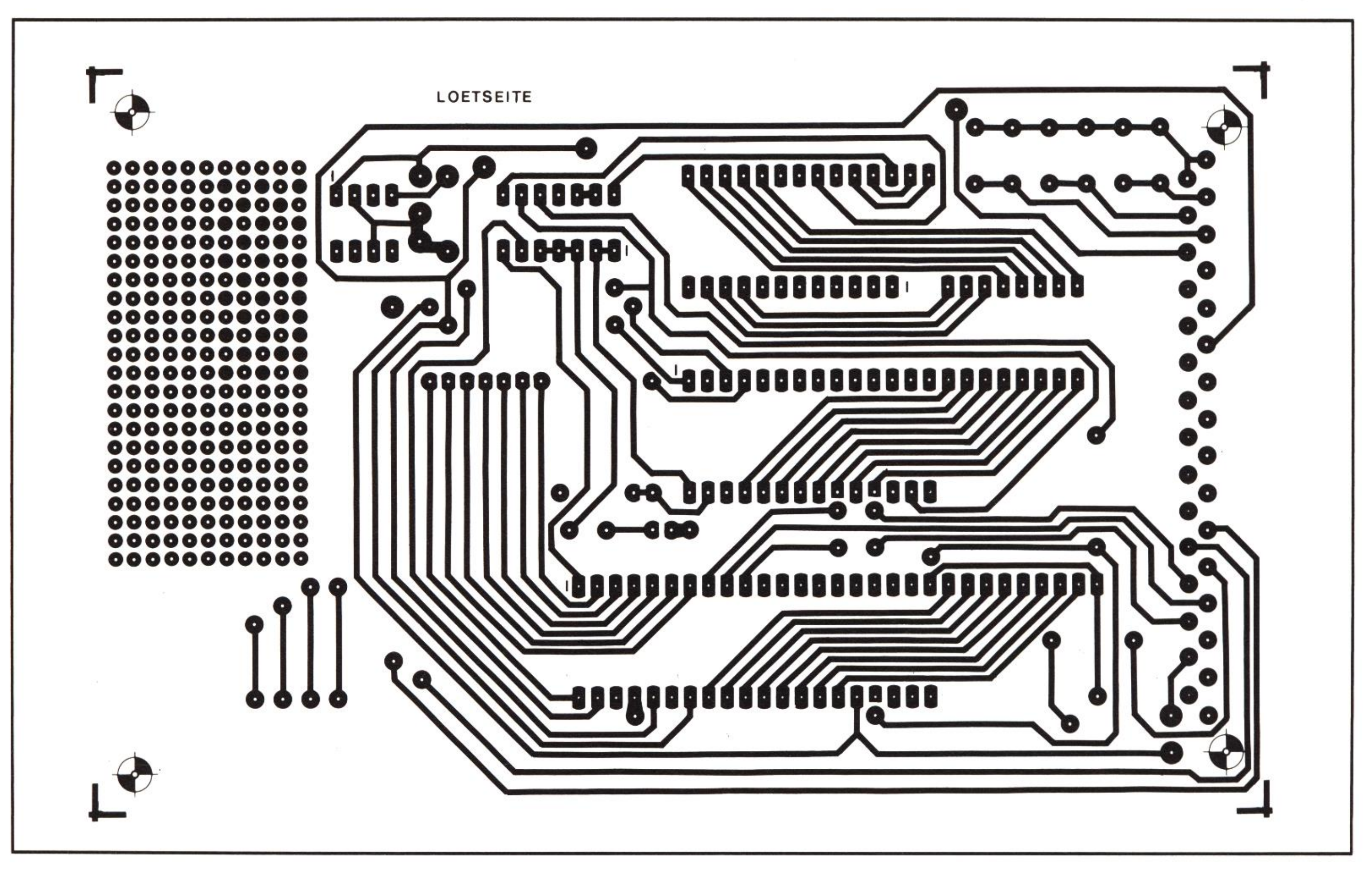


Bild 2. Lötseite der Platine. Sie enthält ein Lochraster-Feld, das vom Anwender für besondere Aufgaben frei verdrahtet werden kann, z. B. für die Nachrüstung eines D/A-Wandlers

Die Drei-Chip-Lösung hat es in sich

Wegen der Verbreitung des Prozessors 6502 bei den preiswerteren Tischcomputern wurde eine CPU aus dieser Familie gewählt, nämlich der Typ 6504. Er unterscheidet sich von der „Mutter" 6502 dadurch, daß er statt 16 nur 12 Adressenleitungen besitzt, nur einen Interrupt-Eingang herausführt ($\overline{IRQ}$), in einem 28-Pin-Gehäuse untergebracht ist (6502: 40 Pins) und nicht zuletzt deshalb auch preiswerter ist.
Wie der geneigte Leser weiß, braucht man in einem Mikrocomputer neben der CPU noch drei Dinge, nämlich einen Arbeitsspeicher (RAM), einen Eingabe/Ausgabe-Baustein (I/O), über den die Verbindung zur Außenwelt hergestellt wird und der somit dafür sorgt, daß der Computer kein Selbstzweck ist, sowie einen Programmspeicher, der hier gemäß dem Verwendungszweck als Festwertspeicher ausgeführt ist.
Um die Chip-Anzahl gering zu halten, findet hier ein Baustein namens 6532 Verwendung, der nicht nur zwei 8-Bit-I/O-Ports sowie 128 Byte RAM enthält, sondern auch einen für mancherlei Zwecke äußerst nützlich programmierbaren Interrupt-Timer, der Zeiten bis zu 261 ms liefern kann. Dazu wird nun nur noch ein EPROM benötigt, das das Betriebsprogramm enthält – in unserem Fall z. B. ein 1-KByte-Typ namens 2758, der ebenfalls schon recht preiswert zu haben ist.

Reicht denn das wirklich aus?

Wenn hier von kläglichen 128 Byte RAM und 1 KByte EPROM die Rede ist, wird manch Tischcomputer-Benutzer sagen, was soll ich damit schon anfangen? Für einen Basic-Computer wäre das tatsächlich viel zu wenig, denn allein ein Basic-Interpreter belegt ja schon rund 4...12 KByte ROM bzw. EPROM. Da Basic aber für die meisten Steuerungszwecke und für zeitkritische Aufgaben völlig ungeeignet ist, wird unser Mikro-Mikrocomputer in der Maschinensprache des verwendeten Prozessors programmiert.
Hier sei gleich vermerkt, daß der 6504 genau den gleichen Befehlssatz wie sein großer Bruder 6502 besitzt und somit zumindest softwaremäßig keinerlei Einschränkungen unterliegt. Und in 1 KByte bringt man z. B. schon ein kleines Schachprogramm unter, ein Programm zur Ansteuerung einer Schreibmaschine über eine serielle Schnittstelle, die Software zum Betrieb eines „dummen" Matrixdruckers oder vieles andere mehr. Übrigens sitzt solch ein 6504-Prozessor auch in der Floppy-Disk-Einheit CBM-3032 von Commodore – auch das ist eine Steueraufgabe, die mit einer Mikrocomputer-Minimalkonfiguration wunderbar zu lösen ist. Also keine Angst vor zu wenig Speicherplatz!

Adressierungs-Kniffe müssen sein

6502-Kenner wissen, daß dieser Prozessor zwei besondere Speicherbereiche besitzt, die beide vorhanden sein müssen, aber hardwaremäßig in ihrer Adressenlage leider mehr als 128 Bytes auseinanderliegen. Unsere 128 Byte zusammen-

Bild 3. Bestückungsseitige Leiterbahnen der (doppelseitigen, durchkontaktierten) Platine. Die 31polige Steckerleiste ist später auf diese Seite zu löten

Bild 4. Bestückungsplan des 6504-Computers. Es sei erwähnt, daß das 2-KByte-EPROM 2716 z. T. schon preiswerter angeboten wird als der 5-V-/1-KByte-Typ 2758. Beim 2716 kann man entweder eine Hälfte „verschenken“ oder auch mit einem Schalter zwischen zwei 1-KByte-Betriebsprogrammen wählen

hängendes RAM würden dafür nicht ausreichen: Wir brauchen einen Bereich in der „Zero Page" (0000...00FF), die nützliche Adressierungsarten bei vielen Maschinensprache-Befehlen des 6502 und die Verwendung speichersparender 2-Byte-Befehle ermöglicht, und einen weiteren in der Page 1 (0100...01FF), der die für Unterprogrammsprünge erforderlichen Rücksprungadressen speichert und gemeinhin als Stack bezeichnet wird.
Dieses Problem wurde hier aber auf eine listige Art umgangen: nämlich mit der sonst mit Recht verpönten Technik, den Adressenbus nicht vollständig zu decodieren und dadurch Speicherplätze scheinbar an mehreren Adressen gleichzeitig erscheinen zu lassen. Und so erscheinen unsere 128 Byte RAM nicht nur an den Zero-Page-Adressen 0000...007F, sondern – mit dem gleichen Speicherinhalt – bei 0180...01FF, also im Stack-Bereich.
Dabei muß man nur bedenken, daß das Schreiben z. B. an die Adresse 01FE den Inhalt bei 007E gleichermaßen verändert. Man muß sich also beim Programmieren überlegen, wieviel Platz man für Unterprogrammsprünge in Stack und wieviele Bytes man in der Zero Page benötigt. Die Verteilung der 128 Bytes RAM könnte dann typischerweise so aussehen, daß 01F0...01FF als Stack dient, um maximal sechs Unterprogrammebenen plus eine Interrupt-Ebene zuzulassen, und 0000...006F als frei verwendbarer Zero-Page-Bereich.
Der anderswo große Nachteil, daß eine Systemerweiterung wegen der unvollständigen Adressendecodierung schwierig ist, wurde hier im Interesse möglichst geringer Hardware-Kosten bewußt inkauf genommen.
Die restliche Adressenbelegung entstand ebenfalls unter diesem Aspekt; es ist nur noch ein einziges TTL-IC nötig, um die Decodierung der Adressen vorzunehmen. Die genaue Zuordnung geht aus *Tabelle 1* hervor.
Die Eigenschaft der Adressenduplizierung kann u. U. auch einen Vorteil darstellen. Denn nicht immer steht in dem Tischcomputer, der zur Entwicklung des Programms Verwendung findet, derjenige Adressenbereich zur Verfügung, in dem der EPROM-Bereich unseres kleinen Systems eigentlich liegt. Möglicherweise besitzt der Tischcomputer aber einen Speicherbereich, der identisch mit einem duplizierten Bereich des EPROM ist. Eine Adressenanpassung ist dann nicht mehr nötig. Dies gilt selbstverständlich auch für die Zero-Page- und Stack-Bereiche.

Tabelle 1: Adressenbelegung des 6504-Computers

Adressenbits	Inhalt	Adressenbereiche
00XX XAAA AAAA	128 Byte RAM im 6532	000...07F; 080...0FF; 100...17F; 180...1FF; 200...27F; 280...2FF; 300...37F; 380...3FF
10XX XXXA AAAA	I/O-Ports und Timer im 6532	800...81F u.a. (32mal dupliziert bis BFF)
11AA AAAA AAAA	EPROM (1 KByte)	C00...FFF
01AA AAAA AAAA	Expansion (1 KByte)	400...7FF

(A = gültiges Adressen-Bit, X = ignoriertes Adressenbit)
6532-Adressen: 800 = Port A, 801 = Port-A-Richtungsregister, 802 = Port B, 803 = Port-B-Richtungsregister; 814 = Timer 1 µs, 815 = Timer 8 µs, 816 = Timer 64 µs, 817 = Timer 1024 µs; 81C...81F wie 814...817, jedoch mit Interrupt bei abgelaufener Zeit. Timer auslesen: 816; Timer testen: 817 (N-Flag).

Die Inbetriebnahme des Systems

Nehmen wir an, wir hätten ein EPROM mit dem nötigen Betriebsprogramm für unseren individuellen Verwendungszweck programmiert. Dann können wir alle Bauelemente auf die doppelseitige durchkontaktierte Epoxy-Platine löten (Bilder 2 bis 4; beziehbar u. a. bei Fa. Walter, Am Starzenbach 9, 8069 Wolnzach), wobei es sich dringend empfiehlt, für die drei LSI-ICs 6504, 6532 und 2758 Fassungen und eine 31polige Steckerleiste (*Tabelle 2*) zu verwenden. Einen Bausatz liefert die Firma Elektronikladen, Wilhelm-Mellies-Str. 88, 4930 Detmold 1.
Beim Anschalten der 5-V-Versorgungsspannung (Netzteil-Belastbarkeit min. 200 mA) erfolgt über das auf der Platine befindliche Monoflop automatisch ein Reset, so daß der Prozessor mit dem Abarbeiten des Programms beginnt, dessen Startadresse in den Zellen FFFC (niederwertiges Byte) und FFFD (höherwertiges Byte) abgelegt ist. Diese Adressen gibt es in unserem System natürlich nicht wirklich; sie finden sich aber dupliziert am oberen Ende des EPROM-Bereichs bei 0FFC und 0FFD.

Tabelle 2: Steckerbelegung

Pin	Belegung
1	Masse
2	Masse
4	IRQ
6	PA0
7	PA1
8	PA2
9	PA7
10	PA6
11	PA5
12	PA4
13	PA3
14	Masse
15	PB0
17	PB1
18	PB2
19	PB3
21	Reset-Ausgang
22	PB7
23	PB6
24	PB5
25	PB4
26	Reset-Eingang
27	+ 5 V
28	− 5 V (bei 2716 + 5 V)
29	+ 12 V (bei 2716 Masse)
30	Masse
31	Masse

Literatur

[1] R 6532 Data Sheet. Rockwell Doc. Nr. 29 000 D42.
[2] R 650X Data Sheet. Rockwell Doc. Nr. 29 000 D39.
[3] R 6500/6532 Timer Interrupt Precautions. Rockwell Doc. Nr. R 6500 N02.
[4] EMUF-Programmiertips. mc 1981, Heft 2.
[5] Bits und Bytes: 6502-Programmierung. Sonderheft „Hobbycomputer 2", Franzis-Verlag.

Otmar Hacker, Mathias Ott

Selbstlernende Haus-Heizungs-regelung

Mit geringem Aufwand an Hard- und Software hält die hier beschriebene Heizungsregelung die Haus-Innentemperatur im Tag- und Nachtabsenk-Betrieb auf ein Grad Celsius genau. Irgendwelche Regler-Einstellungen sind nicht nötig, da sich der Computer – hier ein EMUF – den veränderten Bedingungen wie Haustyp, Wetterlagen usw. selbst anpaßt. Aus der stets richtigen Reglersteuerung und der guten Regeldynamik resultiert eine erhebliche Energie-Ersparnis.

Bei dem verwendeten selbstlernenden, adaptiven System handelt es sich um eine suchalgorithmische Optimierung mit veränderlicher Suchschrittweite. Die Messung und Ausregelung der Temperaturen erfolgt mit einer Abtastregelung.

Das Programm wurde zunächst mit einem PC-100 (alias AIM-65) entwickelt und getestet, später aber auf den Einplatinen-Computer EMUF übertragen, den mc 1981 in Heft 2 sowie im EMUF-Sonderheft ausführlich beschrieb und der weniger als 100 DM kostet.

Heizungsregelungen: Ein wenig Theorie

Herkömmliche Haus-Heizungsregelungen verstellen bei Konstanttemperatur-Kesseln die Heißwasser- oder Vorlauftemperatur T_v mit einem Mischerventil (*Bild 1*). Das geschieht abhängig von der Außentemperatur T_a nach gekrümmten Kurven. Die Haus-Innentemperaturen sollen dadurch annähernd auf einem konstanten Wert (z. B. 20 °C) gehalten werden.
Dazu muß die Regelschaltung mit bis zu fünf Einstellknöpfen an das Haus angepaßt werden. Eine wirklich exakte Temperaturregelung ist so nur schwer zu erreichen und ein Überschwingen der Raumtemperaturen kaum zu vermeiden. Denn die nötige Vorlauftemperatur hängt nicht nur von der Außentemperatur, sondern auch von anderen Witterungseinflüssen wie Windstärke oder Luftfeuchtigkeit ab. Die nicht exakte Einstellung führt auch zu einem unnötig hohen Energieverbrauch.
Die adaptierende Regelung dagegen nutzt die Möglichkeiten der Mikrocomputer-Intelligenz, um mit Hilfe ausgeklügelter Programme derartige Anpassungsarbeiten zu ersparen. Wie *Bild 2* zeigt, wird ein Pendeln oder Überschwingen der Temperaturen insbesondere beim Wiederaufheizen am Morgen

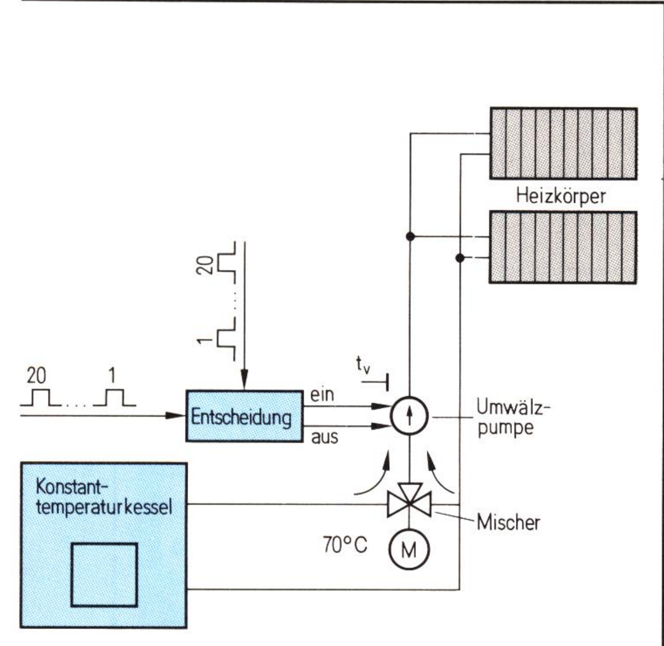


Bild 1. Anordnung einer Heizungsanlage mit Konstanttemperatur-Kessel. Die Vorlauftemperatur, also die Temperatur des Wassers in den Heizkörpern, wird über den Mischer gesteuert

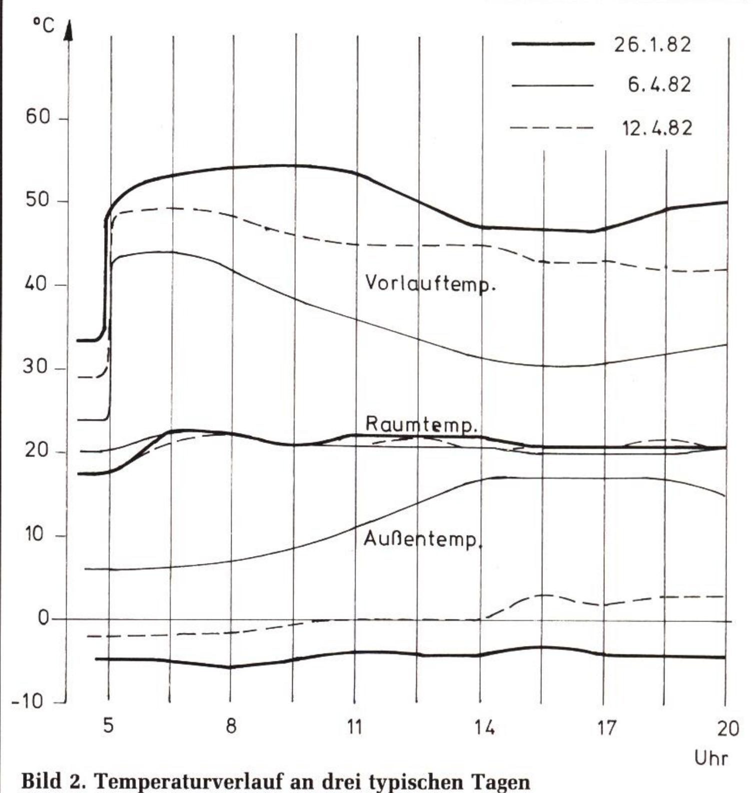


Bild 2. Temperaturverlauf an drei typischen Tagen

vermieden. Auch ein Nachregeln durch Thermostat-Ventile an den Heizkörpern ist nicht nötig; allerdings können solche Ventile Einflüsse wie direkte Sonneneinstrahlung an der Südseite ausgleichen.

Das adaptive Regelsystem

Bei unserer selbstlernenden Regelung handelt es sich im Prinzip um eine Nachlauf-Regelung der Heizwasser-Vorlauftemperatur T_v zu einer per Fühler gemessenen Außentemperatur T_a als Führungsgröße. Die Messung der Außentemperatur ist nötig, weil die unmittelbare Regelung der Innentemperatur wegen der großen Verzögerung verspätet korrigiert würde: Überschwingen und Pendeln wäre die Folge.

Unter Abtastregelung versteht man, daß die einzelnen Temperaturfühler stets nacheinander erfaßt und verarbeitet werden, wie es typischerweise in einem Computer geschieht.

Der Computer ermittelt die optimale Kurve der Regelung selbst durch einen Suchalgorithmus. Je kleiner die Temperaturabweichungen werden, desto kleiner werden auch die Suchschritte, so daß mit der Zeit eine immer bessere Anpassung an die tatsächlichen Verhältnisse stattfindet.

Bild 3 zeigt die dafür nötige Schaltung. Der Hardware-Aufwand für das selbstlernende System ist erstaunlich gering: hier ein EMUF (Einplatinen-Mikrocomputer für universelle Festprogramm-Anwendung) mit 1 KByte EPROM und einem 6504-Prozessor. Zur Programmentwicklung diente ein Tischcomputer PC-100 mit Assembler.

Trickreiche Temperaturmessung

Eine besonders wirtschaftliche Realisation einer Temperaturmessung per Computer ergibt sich durch die Anwendung eines Tacho-Bausteins (LM 2907). Herkömmliche D/A-Wandler würden dafür mindestens zehn Rechneranschlüsse benötigen; der Tachobaustein kommt mit zweien aus.

Der Computer gibt eine Frequenz aus, die per Programm erzeugt wird, und der Tachobaustein macht daraus eine der Frequenz proportionale Spannung. Ein Operationsverstärker vergleicht diese mit dem Ausgangssignal eines Temperaturfühlers und leitet das Ergebnis wieder dem Computer zu.

Der Rechner „wobbelt" zunächst über ein relativ breites Frequenzband. Sobald der Operationsverstärker im LM 2907 ihm meldet, daß die der gemessenen Temperatur entsprechende Frequenz erreicht ist, schaltet er auf ein Frequenzband um, das nur wenig unter dieser beginnt. Dadurch können die Temperaturen dann wesentlich schneller gemessen werden. Vorlauf- und Außentemperaturen werden zur Sicherheit mehrmals gemessen; der Regelzyklus beträgt 90 Sekunden. Über die Pulslänge wird der Mischermotor proportional geregelt, ein weiterer Computer-Ausgang schaltet seine Drehrichtung.

Die Innentemperatur wird wegen der großen Haus-Zeitkonstanten nur etwa alle 90 Minuten gemessen – ebenfalls mehrmals hintereinander, um Falschwerte auszuschalten, die z. B. durch Störimpulse entstehen könnten. Auch wird ein zu häufiges Motor-Ein- und -Ausschalten vermieden.

Temperaturabsenkung nachts

Eine externe, netzsynchrone Schaltuhr, wie sie im Handel preisgünstig zu haben

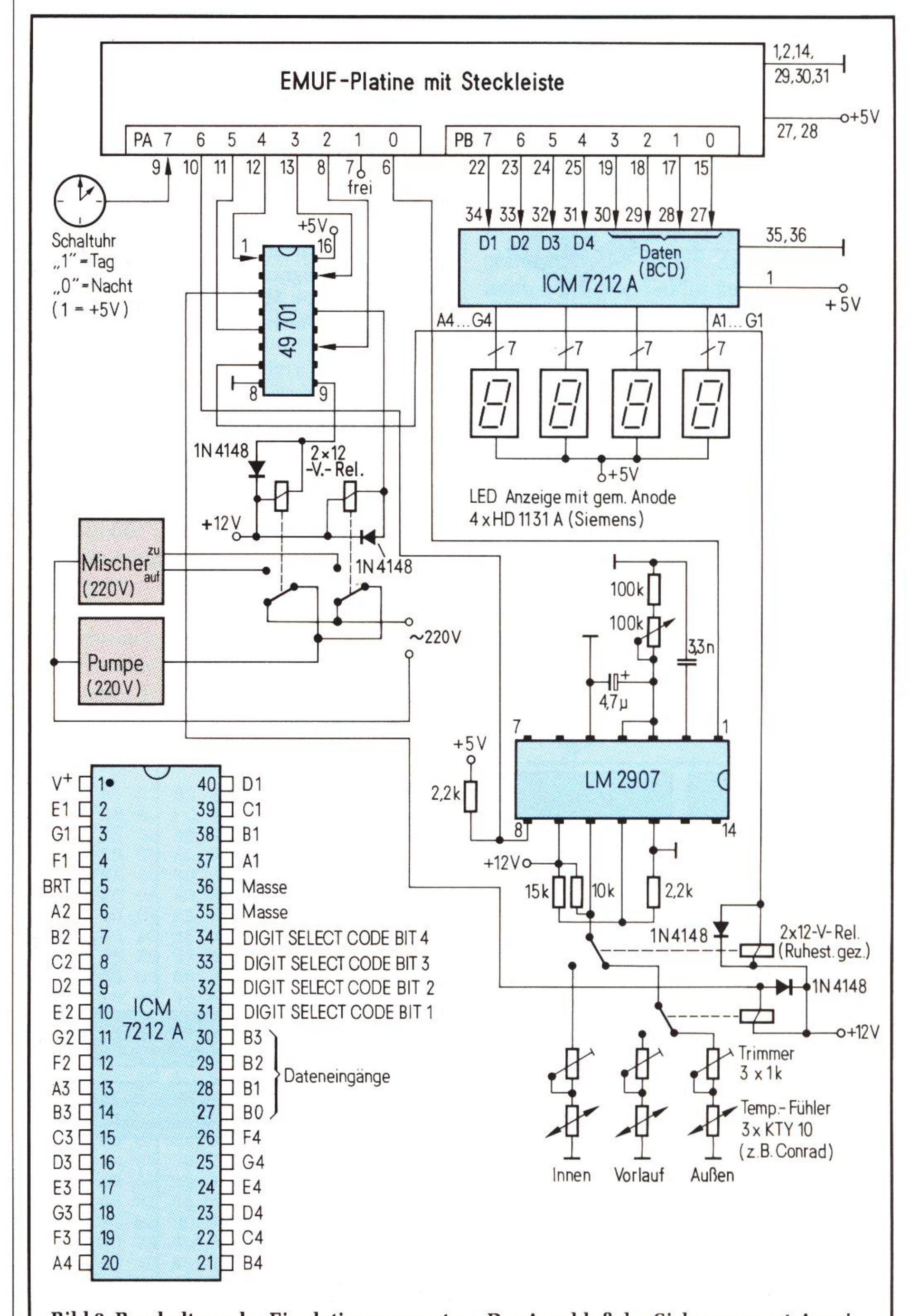


Bild 3. Beschaltung des Einplatinencomputers. Der Anschluß der Siebensegment-Anzeigen geht aus der Pinbelegung des Display-Controllers ICM 7212 A hervor

ist, gibt am Abend einen Impuls an den Computer ab. Dieser verschiebt die Regelgerade nach unten; die Regelung selbst bleibt jedoch wie tagsüber wirksam. Allerdings wird die eingestellte Heizkurve nicht mehr verändert. Am Morgen verschiebt ein weiteres Signal der Uhr die Gerade auf den letzten Stand vor der Nachtabsenkung. Bei Abwei-

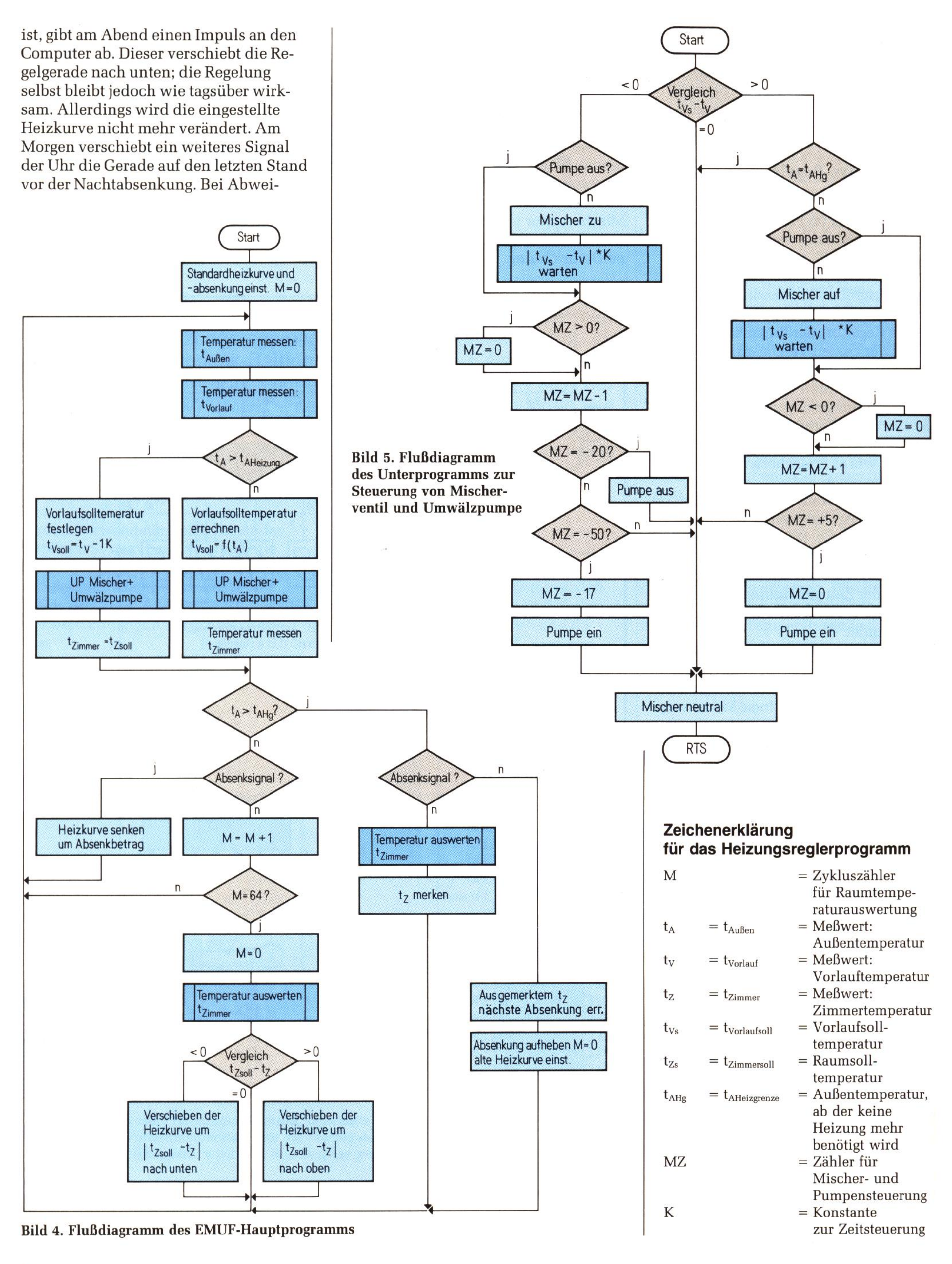


Bild 4. Flußdiagramm des EMUF-Hauptprogramms

Bild 5. Flußdiagramm des Unterprogramms zur Steuerung von Mischerventil und Umwälzpumpe

Zeichenerklärung für das Heizungsreglerprogramm

M		= Zykluszähler für Raumtemperaturauswertung
t_A	$= t_{Außen}$	= Meßwert: Außentemperatur
t_V	$= t_{Vorlauf}$	= Meßwert: Vorlauftemperatur
t_Z	$= t_{Zimmer}$	= Meßwert: Zimmertemperatur
t_{Vs}	$= t_{Vorlaufsoll}$	= Vorlaufsoll-temperatur
t_{Zs}	$= t_{Zimmersoll}$	= Raumsoll-temperatur
t_{AHg}	$= t_{AHeizgrenze}$	= Außentemperatur, ab der keine Heizung mehr benötigt wird
MZ		= Zähler für Mischer- und Pumpensteuerung
K		= Konstante zur Zeitsteuerung

chungen der Innentemperatur merkt sich der Rechner für die nächste Nachtabsenkung eine entsprechende weitere kleine Verschiebung der Regelgeraden vor. Diese Anpassung mittels Suchalgorithmus kann sich einige Tage wiederholen, bis eine optimale Regelung erreicht ist.

Bei steigenden Außentemperaturen, z. B. über 14 °C, mußte bisher die Umwälzpumpe noch von Hand abgeschaltet werden, ebenso nachts. Dies besorgt jetzt der Rechner ohne zusätzlichen Hardware-Aufwand, denn der Pumpenschalter wird von den vier Relais-Verstärkern mit zwei Rechnerausgängen mitgesteuert. Die Pumpe wird abgeschaltet, nachdem der Mischer 20 Schließbefehle erhalten hat.

Vor dem Abschalten ist sichergestellt, daß das Mischerventil völlig zu ist, um schädliche „Schwerkraftzirkulation" im Heißwasserkreislauf zu vermeiden. Gleichzeitig werden „Auf-Befehle" unterdrückt, um ein Pendeln des Ventiles zu vermeiden. Bei stehender Pumpe wird die Innentemperatur gemessen, aber nicht mehr zur Adaption benutzt. Damit verhindert man, daß bei erhöhten Temperaturen infolge Sonneneinstrahlung z. B. die T_v-T_a-Gerade falsch adaptiert wird. Außerdem wird die Pumpe im Stundenzyklus 2 bis 5 Minuten eingeschaltet. Damit erfaßt der Vorlauftemperatur-Fühler auch die tatsächliche Temperatur des Wassers und Einfrieren wird verhindert. Bei stehender Heizungspumpe werden die Mischerbefehle zunächst unterdrückt. Die Pumpe wird eingeschaltet, nachdem fünf Mischer-Auf-Befehle anstanden.
Die LED-Anzeigen zeigen die drei Temperaturen in °C als 3. und 4. Stelle. Die 2. Stelle bedeutet mit „–" negative, fehlende Anzeige positive Temperaturen. Die 1. Stelle (links) signalisiert mit „1", „2" und „3" die Außen-, Vorlauf- und die Raumtemperaturen im Tagesbetrieb mit „4", „5" und „6" den Nachtabsenkungsbetrieb der zwei rechten Stellen.
Mit Hilfe der drei Trimmwiderstände werden die Temperaturwiderstände am Display eingestellt bzw. kontrolliert.
Die *Bilder* 4 und 5 geben die Flußdiagramme des Hauptprogramms und des Unterprogramms zur Steuerung des Mischerventils und der Vorlaufpumpe wieder. In der *Tabelle* finden sich die dabei verwendeten Abkürzungen für die Programm-Parameter.
Bild 6 zeigt das fertige Maschinenprogramm für den EMUF. Zusätzlich ist noch der Reset-Vektor zu setzen: 0FFC = 00, 0FFD = 0C. Sinnvollerweise überprüft man das Programm vor dem Brennen in ein EPROM mit Prüfsummen. Diese betragen (jeweils von Anfangsadresse bis Endadresse−1 aufsummiert):

6F38 für 0C00...0D00,
674C für 0D00...0E00,
801E für 0E00...0F00 und
6213 für 0C00...0F28.

Ein Prüfsummenprogramm für den als Entwicklungssystem geeigneten AIM-65 oder PC-100 findet sich in mc 1981, Heft 2, auf Seite 36, oder in mc 1982, Heft 5, Seite 55.

Antiquarisches

... Aber ich sage, daß diejenigen Psychologen, die zwischen der gefühlsbetonten Handlung des Menschen und anderer Lebewesen und der Handlung des modernen Typs von automatischen Mechanismen scharfe und prinzipielle Unterschiede machen, ebenso vorsichtig bei ihren Einwänden sein sollen wie ich bei meinen Behauptungen.

Norbert Wiener, im Buch „Mensch und Menschmaschinen", geschrieben 1949

```
0C00 A2 FF 9A D8 86 3A 86 3B A9 F0 85 11 85 21 85 31
0C10 A9 3F 8D 01 08 EA EA EA EA A9 80 85 38 A9 F0 85
0C20 36 A9 00 85 35 A9 FF 8D 03 08 A9 00 85 32 A9 00
0C30 85 33 85 34 A5 21 85 41 A9 FF EA 20 59 0D A5 41
0C40 D0 02 A9 CC 85 21 A5 11 85 41 A9 EF EA 20 59 0D
0C50 A5 41 D0 02 A9 90 85 11 A5 21 C9 C9 10 0A A9 FF
0C60 20 1F 0E A9 30 4C 7E 0C 38 65 11 20 1C 0E A5 31
0C70 85 41 A9 DF EA 20 59 0D A5 41 D0 0D A9 10 85 32
0C80 A9 00 85 33 85 34 EA A9 C1 85 31 A5 35 F0 57 AD
0C90 00 08 29 80 D0 26 A5 32 29 08 F0 10 A0 03 66 33
0CA0 66 34 88 D0 F9 A5 34 85 37 4C 2A 0C 18 A5 34 65
0CB0 31 85 34 A5 33 69 00 85 33 4C 0E 0D A9 00 85 35
0CC0 38 A5 36 65 38 85 38 A5 37 38 E9 C6 18 65 36 C9
0CD0 EA 10 02 A9 EA 85 36 EA EA EA EA EA EA EA EA EA
0CE0 EA A9 09 4C 2C 0C A5 32 C9 08 F0 2D 29 F8 F0 59
0CF0 AD 00 08 29 80 D0 17 A9 01 85 35 A5 36 49 FF 18
0D00 65 38 85 38 EA EA EA EA EA EA EA 4C 2A 0C E6 32
0D10 A5 32 29 3F 85 32 4C 34 0C A0 03 66 33 66 34 88
0D20 D0 F9 EA EA EA A5 34 38 E9 C2 49 FF 18 65 38 C9
0D30 7A 10 05 A9 7A 4C 3E 0D C9 86 30 02 A9 86 85 38
0D40 A9 00 85 33 85 34 4C 0E 0D 18 A5 34 65 31 85 34
0D50 A5 33 69 00 85 33 4C 0E 0D 25 3A 85 30 8D 00 08
0D60 A9 40 85 1A A9 00 85 42 85 43 AD 00 08 29 40 F0
0D70 F9 A2 06 E6 41 F0 2D CA D0 F9 A6 41 A0 30 A5 30
0D80 49 01 85 30 8D 00 08 86 10 C6 10 D0 FC A9 10 85
0D90 09 C6 09 D0 FC AD 00 08 29 40 F0 11 88 D0 DF CA
0DA0 D0 DA 86 41 A9 0A 85 14 4C 04 0E EA EA 86 41 18
0DB0 A5 42 65 41 85 42 A5 43 69 00 85 43 C6 1A D0 AA
0DC0 26 42 26 43 26 42 26 43 A9 00 85 14 A5 43 85 41
0DD0 38 E9 D6 85 20 10 0A A2 0F 38 A9 00 E5 20 4C E3
0DE0 0D A2 0A 8E 02 08 A8 8A 09 40 8D 02 08 98 C9 64
0DF0 30 06 38 E9 64 4C EE 0D C9 0A 30 08 38 E9 0A E6
0E00 14 4C F8 0D 8D 02 08 09 10 8D 02 08 A5 14 8D 02
0E10 08 09 20 8D 02 08 20 00 0F 60 EA EA 38 E5 38 85
0E20 1B 30 41 A5 21 C9 C9 F0 37 A5 3A 29 0C F0 19 A9
0E30 FB 8D 00 08 A5 1B D0 08 A9 FF 8D 00 08 4C 48 0E
0E40 20 B0 0E C6 1B 4C 36 0E A5 3B 10 04 A9 00 85 3B
0E50 E6 3B A5 3B C9 05 D0 08 A9 FF 85 3A A9 00 85 3B
0E60 60 EA EA EA A5 3A 29 0C F0 19 A9 F7 8D 00 08 A5
0E70 1B D0 08 A9 FF 8D 00 08 4C 83 0E 20 B0 0E E6 1B
0E80 4C 71 0E A5 3B 30 04 A9 00 85 3B C6 3B A5 3B C9
0E90 EC D0 05 A9 F3 85 3A 60 C9 BA D0 08 A9 FF 85 3A
0EA0 A9 EF 85 3B 60 A1 E6 A4 A7 8C AF 1B AF E2 1E 8F
0EB0 A5 1F 45 1B 29 80 F0 03 20 BB 0E A5 1B 85 1F A9
0EC0 0A 85 1C A9 FF 85 1D A9 FF 85 1E C6 1E D0 FC C6
0ED0 1D D0 F4 C6 1C D0 EC 60 EA EA EA EA EA EA EA EA
0EE0 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA
0EF0 EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA EA
0F00 A5 30 29 20 F0 10 A5 30 29 10 F0 05 A9 01 4C 18
0F10 0F A9 02 4C 18 0F A9 03 A6 35 F0 03 18 69 03 8D
0F20 02 08 09 80 8D 02 08 60 FF F0 11 A9 0E 85 FD 88
0FFC 00 0C
```

Bild 6. Hex-Dump des EPROM-Inhaltes

Anton Ruepp

EMUF als serielles Interface

Drucker am seriellen Commodore-Bus

Genaugenommen ist diese Applikation für alle Computer verwendbar, die den seriellen (IEC-ähnlichen) Bus besitzen, also VC-20, C-64 und Nachfolger. Mit EMUF-Unterstützung lassen sich nun auch Drucker mit anderen Schnittstellen anschließen, die aber rechnerseitig ohne zusätzliche Software mit den normalen Befehlen angesprochen werden können. Das vorliegende Beispiel zeigt die Lösung für eine serielle 20-mA-Schnittstelle, die leicht in eine V.24-Schnittstelle umgewandelt werden kann.

Der EMUF [1] bedient die serielle Schnittstelle des Computers und gibt die empfangenen Zeichen in der gewünschten Form an den angeschlossenen Drukker aus. Hierbei lassen sich auch eventuelle Zeichenumcodierungen oder Sonderzeichen realisieren. Im vorliegenden Fall handelt es sich um eine TTY-Schnittstelle, also eine 20-mA-Schleife, mit einer Übertragungsrate von 150 Baud und ungerader Parität.
Es lassen sich natürlich auch andere Schnittstellen realisieren. Dazu ist entweder nur eine Hardware-Anpassung erforderlich (V.24), oder aber das Zeichenausgabe-Programm muß umgeschrieben werden (Centronics).

Programmablauf

Nach der Initialisierung wartet der EMUF auf seine Primäradresse, die im Falle des Druckers 4 lautet. Sobald er diese erkannt hat, fühlt er sich so lange angesprochen, bis der Rechner „Unlisten“ ausgibt. Sekundäradressen werden ignoriert. Eine ausführliche Beschreibung des seriellen Busses findet sich in [2].
Die Routine GBYTE des Programmes (*Bild 1*) liest ein Zeichen vom Bus und legt es in Zero-Page in der Zelle SHIFT ab, die bei der Ausgabe als Schieberegister verwendet wird. Bei Empfang des Unlisten-Zeichens erfolgt ein Neustart. Die Routine AUSGA fügt bei Erkennen eines Carriage Return (CR) noch ein Linefeed (LF) ein, wenn der Pegel an PA0 dies verlangt. Das Programm OUTALL

```
0000                      *=0
0000             PARYB    *=*+1             ;PARITY-ZAEHLER
0001             TIMER    *=*+1             ;EOI TIME REGISTER
0002             SHIFT    *=*+1             ;SCHIEBEREGISTER
0003             OUTBD    *=*+1             ;ZAEHLER FUER BAUD GENER
0004             ATNFG    *=*+1             ;ATNFLAG,TRUE (LOW) ==> $FF
0005             PRTA=$800
0005             DDRA=$801
0005             PRTB     =$802
0005             DDRB     =$803
0005             CLOCK    =%00100000
0005             ;
0005                      *=$FFC            ;RESET
0FFC  00 0C               .WOR START
0FFE                      *=$FFE            ;IRQ
0FFE  00 0C               .WOR START        ;UNUSED
1000             ;
1000                      *=$C00
0C00  78         START    SEI               ;DO NOT DISTURB
0C01  A2 FF               LDX #$FF          ;ORDNE
0C03  9A                  TXS               ;STACK
0C04  D8                  CLD               ;NO DEC
0C05             ;
0C05  20 36 0C            JSR INIT          ;INITIALISIERUNG
0C08  20 54 0C            JSR ATNHI         ;WARTE BIS ATN NEUTRAL IST
0C0B  20 4A 0C   HAUPT    JSR ATNLO         ;UND JETZT BIS ATN TRUE.
0C0E  AD 00 08            LDA PRTA          ;WENN CLOCK HI
0C11  29 20               AND #%00100000    ;IST, VERSUCHEN WIR'S
0C13  D0 F6               BNE HAUPT         ;NOCHMALS
0C15  20 62 0C            JSR SDLOW         ;SETZE DATA LOW
0C18  20 5A 0C            JSR CLOHI         ;UND WARTE AUF CLOCK HI
0C1B  20 8A 0C            JSR GBYTE         ;HOL DAS ZEICHEN
0C1E  20 70 0C            JSR MYADS         ;BIN ICH GEMEINT ?
0C21  B0 DD               BCS START         ;NEIN
0C23  20 8A 0C   MYLA     JSR GBYTE         ;EIN BYTE VOM BUS HOLEN
0C26  24 04               BIT ATNFG         ;WAR ES ETWA
0C28  F0 06               BEQ CONT          ;UNTER ATN ?
0C2A  A5 02               LDA SHIFT         ;WENN JA,  -WAR ES
0C2C  C9 3F               CMP #$3F          ;VIELLEICHT "UNLISTEN" ?
0C2E  F0 D0               BEQ START         ;JA. ALSO SCHLUSS
0C30  20 D9 0C   CONT     JSR AUSGA         ;ZEICHEN AUSGEBEN
0C33  4C 23 0C            JMP MYLA          ;NAECHSTES HOLEN
0C36             ;
0C36  A9 00      INIT     LDA #0
0C38  8D 01 08            STA DDRA          ;ALLES INPUTS
0C3B  A9 01               LDA #1
0C3D  8D 02 08            STA PRTB          ;PRINTERSTROM
0C40  A9 FF               LDA #$FF
0C42  8D 03 08            STA DDRB
0C45  A9 01               LDA #1            ;ODD PARITY
0C47  85 00               STA PARYB
0C49  60                  RTS
0C4A             ;WARTET AUF ATN UND CLOCK TRUE
0C4A  AD 00 08   ATNLO    LDA PRTA
0C4D  30 FB               BMI ATNLO
0C4F  29 20               AND #CLOCK
0C51  D0 F7               BNE ATNLO
0C53  60                  RTS
```

Bild 1. Das EMUF-Programm im Quellenformat. Änderungen für andere Baudraten sind in der Tabelle zu finden

```
0C54  AD 00 08    ATNHI  LDA PRTA         ;WARTET AUF ATN HI
0C57  10 FB              BPL ATNHI
0C59  60                 RTS
0C5A                     ;WARTET BIS CLOCK HIGH IST
0C5A  AD 00 08    CLOHI  LDA PRTA
0C5D  29 20              AND #CLOCK
0C5F  F0 F9              BEO CLOHI
0C61  60                 RTS
0C62                     ;SET DATA LOW
0C62  A9 00       SDLOW  LDA #0
0C64  8D 00 08           STA PRTA
0C67  AD 01 08           LDA DDRA
0C6A  09 40              ORA #%01000000   ;PA6 AUSG
0C6C  8D 01 08           STA DDRA
0C6F  60                 RTS
0C70                     ;CLEAR CARRY WENN ADR. OK
0C70  A5 02       MYADS  LDA SHIFT
0C72  C9 24              CMP #$24         ;PRIMAERADRESSE  4
0C74  18                 CLC
0C75  F0 01              BEQ FINI
0C77  38                 SEC
0C78  60          FINI   RTS
0C79                     ;WARTEN AUF CLOCK LOW
0C79  AD 00 08    CLOLO  LDA PRTA
0C7C  29 20              AND #CLOCK
0C7E  D0 F9              BNE CLOLO
0C80  60                 RTS
0C81  AD 01 08    DAINP  LDA DDRA         ;PA6 INPUT (=HIGH)
0C84  29 BF              AND #%10111111
0C86  8D 01 08           STA DDRA
0C89  60                 RTS
0C8A                     ;HOLT EIN BYTE VOM BUS. GIBT EOI
0C8A                     ;HANDSHAKE WENN NOETIG.
0C8A                     ;SETZT ATN-FLAG ENTSPRECHEND (ATN. TRUE= $FF)
0C8A  20 5A 0C    GBYTE  JSR CLOHI        ;WARTE BIS CLOCK HIGH
0C8D  20 81 0C           JSR DAINP        ;MACH PA6 INPUT & HIGH
0C90  A9 0F              LDA #$0F
0C92  85 01              STA TIMER        ; "TIMER" FUER EOI ACK
0C94  AD 00 08    GBY1   LDA PRTA
0C97  29 20              AND #CLOCK
0C99  F0 0F              BEQ CL           ;CLOCK LOW , KEIN EOI
0C9B  C6 01              DEC TIMER
0C9D  D0 F5              BNE GBY1         ;NOCH KEIN EOI
0C9F  20 62 0C    EOI    JSR SDLOW        ;QUITTIERE EOI MIT DATA LOW
0CA2  A2 0C              LDX #12
0CA4  CA          D1     DEX
0CA5  D0 FD              BNE D1
0CA7  20 81 0C           JSR DAINP        ;DATA WIEDER HIGH
0CAA  A2 08       CL     LDX #8
0CAC  20 79 0C    CL2    JSR CLOLO
0CAF  20 5A 0C           JSR CLOHI
0CB2  AD 00 08           LDA PRTA
0CB5  29 40              AND #%01000000   ;DATA
0CB7  0A                 ASL A
0CB8  0A                 ASL A            ;DATA IN CARRY
0CB9  66 02              ROR SHIFT        ;CARRY INS SCHIEBEREGISTER
0CBB  CA                 DEX
0CBC  D0 EE              BNE CL2
0CBE  AD 00 08           LDA PRTA
0CC1  30 04              BMI NIXAT
0CC3  A9 FF              LDA #$FF
0CC5  D0 02              BNE ATNJN        ;JA, ATN
0CC7  A9 00       NIXAT  LDA #00
0CC9  85 04       ATNJN  STA ATNFG
```

```
0CCB  20 79 0C           JSR CLOLO        ;EIN BYTE IST IM KASTEN
0CCE  A9 00              LDA #0
0CD0  8D 00 08           STA PRTA
0CD3  A9 40              LDA #%01000000
0CD5  8D 01 08           STA DDRA         ;DATA ACCEPTED
0CD8  60                 RTS
0CD9                     ; HIER WERDEN ALLE UNTER ATN ANKOMMENDEN
0CD9                     ; ZEICHEN AUSGEFILTERT
0CD9  A5 04       AUSGA  LDA ATNFG
0CDB  D0 18              BNE ENT          ;EXIT WENN ATN FLAG CLEAR
0CDD  A5 02              LDA SHIFT        ;HOLE BYTE
0CDF  29 7F              AND #$7F
0CE1  C9 0D              CMP #$0D         ;CARRIAGE RET
0CE3  D0 0D              BNE NOL          ;NEIN, GIBS AUS
0CE5  48                 PHA              ;SAVE IT
0CE6  AD 00 08           LDA PRTA         ;WIE STEHT DER
0CE9  4A                 LSR A            ;CR-CRLF SWITCH ?
0CEA  90 05              BCC NOLF         ;KEIN ZUSAETZLICHER LF
0CEC  A9 0A              LDA #$0A         ;LF
0CEE  20 F6 0C           JSR OUTALL       ;AUSGEBEN
0CF1  68          NOLF   PLA              ;ORIG. CHAR. HOLEN
0CF2  20 F6 0C    NOL    JSR OUTALL
0CF5  60          ENT    RTS
0CF6                     ;ZEICHEN SERIELL AUSGEBEN
0CF6  A2 07       OUTALL LDX #$07         ;LADE ZAEHLER FUER 7 BIT
0CF8  18                 CLC
0CF9  48                 PHA
0CFA  A9 00              LDA #0           ;SENDE STARTBIT
0CFC  8D 02 08           STA PRTB
0CFF  20 2A 0D           JSR DELAY
0D02  68                 PLA
0D03  8D 02 08    QQQ    STA PRTB         ;SENDE DATENBIT
0D06  4A                 LSR A            ;SCHIEBE DAS NAECHSTE HEREIN
0D07  48                 PHA
0D08  20 2A 0D           JSR DELAY
0D0B  A5 00              LDA PARYB
0D0D  69 00              ADC #0           ;ADDIERE GESENDETES ZU PARY
0D0F  85 00              STA PARYB
0D11  68                 PLA
0D12  CA                 DEX
0D13  D0 EE              BNE QQQ          ;SCHON ALLE GESENDET ?
0D15  A5 00              LDA PARYB
0D17  8D 02 08           STA PRTB         ;GIB PARITY AUS
0D1A  20 2A 0D           JSR DELAY
0D1D  A9 01              LDA #1           ;ODD PARITY
0D1F  85 00              STA PARYB        ;WIEDER INIT
0D21  A9 FF              LDA #$FF
0D23  8D 02 08           STA PRTB         ;SENDE 1 STOPPBIT
0D26  20 2A 0D           JSR DELAY
0D29  60                 RTS
0D2A                     ;BAUDRATENGENERATOR
0D2A                     ;DELAY, VERZOEGERUNGSSCHLAUFE FUER
0D2A                     ; 6,67 MS BITDAUER (1/BAUDRATE)
0D2A                     ; =150 BD
0D2A  A0 10       DELAY  LDY #$10         ;NUR INNERHALB EINER PAGE
0D2C  A9 33       DLY1   LDA #$33         ;OUT - BAUD
0D2E  85 03              STA OUTBD
0D30  C6 03       DLY2   DEC OUTBD
0D32  D0 FC              BNE DLY2
0D34  88                 DEY
0D35  D0 F5              BNE DLY1
0D37  60                 RTS
0D38
0D38                      END
```

Änderungen für verschiedene Baudraten und Paritätsprüfung

Baudrate	Zeit in ms	$0D2B	$0D2D
50	20	10	9A
75	13,3	10	66
100	10	10	4D
150	6,7	10	33
300	3,3	10	19
Paritätsprüfung		$0D1E/0C3C	
Gerade		00	
Ungerade		01	

gibt die Zeichen seriell aus, wobei DELAY den Takt bestimmt. Eine Aufstellung für verschiedene Baudraten zeigt die *Tabelle*.

Etwas zusätzliche Hardware ist vonnöten

Bild 2 zeigt, daß wirklich sehr wenig Hardware zusätzlich zum EMUF erforderlich ist. Sie läßt sich sehr leicht auf dem Lochrasterfeld unterbringen. An PB0 liegt ein Darlington-Transistor, um die 20-mA-Stromschleife zu steuern. Der Schalter an PA0 schaltet die Auto-Linefeed-Funktion ein und aus.

Literatur

[1] *Feichtinger, H.*: Mädchen für alles. mc-EMUF-Sonderheft, Seite 10.

[2] *Löhr, R.*: IEC: Die seriellen Busroutinen. 65XX Micro Mag Nr. 35, Februar 1985, Seite 30.

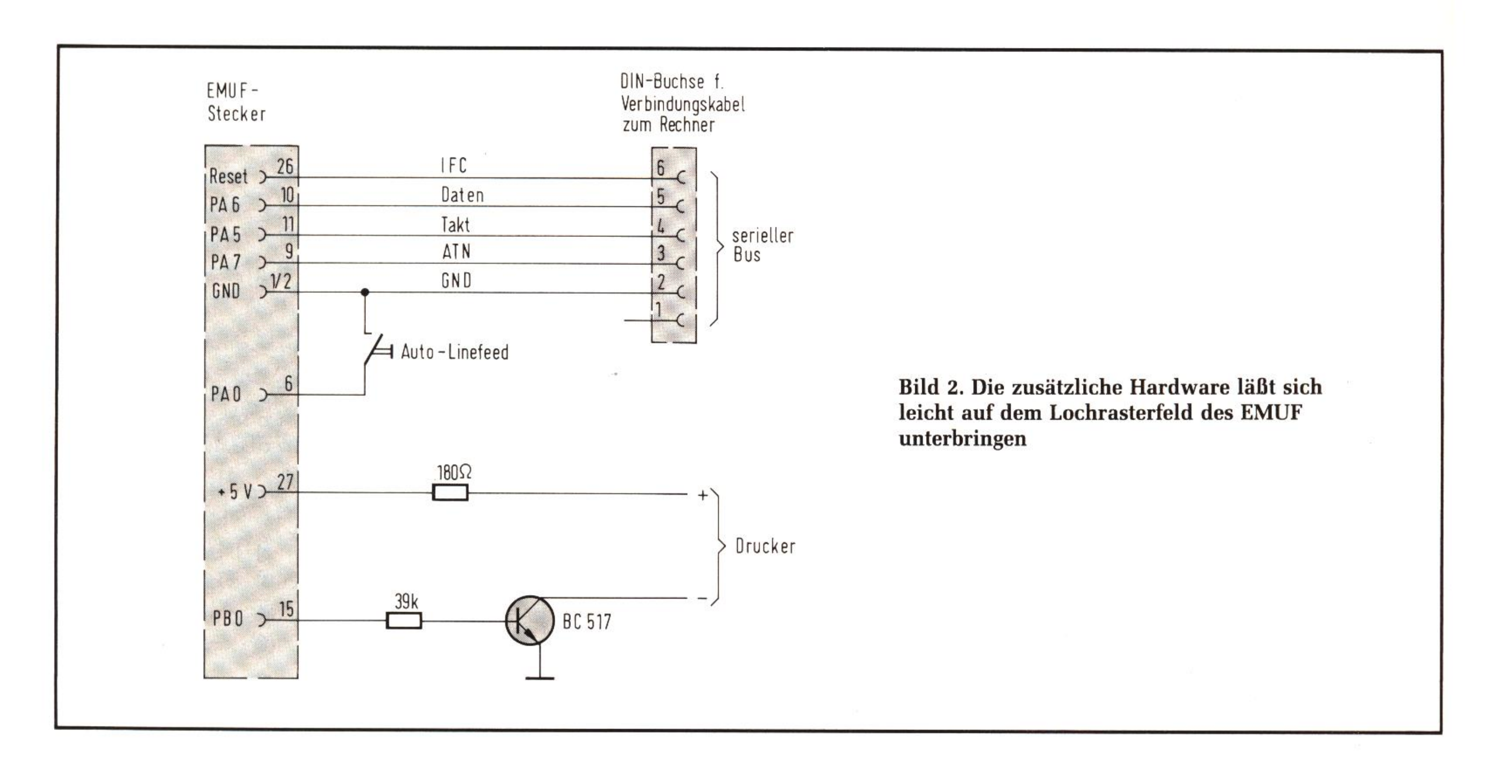
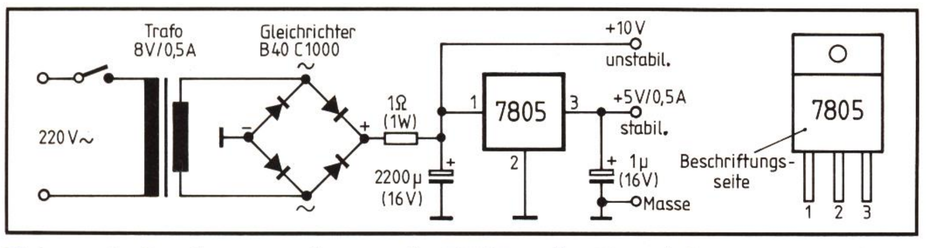


Bild 2. Die zusätzliche Hardware läßt sich leicht auf dem Lochrasterfeld des EMUF unterbringen

Ein Netzteil für den EMUF

Wenn man den Einplatinen-Computer EMUF mit einem EPROM-Typ bestückt, der mit nur einer Versorgungsspannung (+ 5 V) auskommt, so genügt für das ganze Gerät eine einfache Stromversorgung: Die EMUF-Platine selbst benötigt rund 250 mA, und wenn man noch einige Leuchtdioden und zusätzliche ICs betreiben möchte, so dimensioniert man das Netzteil am besten für 5 V/0,5 A. Das *Bild* zeigt eine hierfür geeignete Schaltung. Das Regel-IC 7805 sollte man auf eine Kühlfläche von wenigstens 5 × 5 cm² setzen; ein Rippenkühlkörper ist natürlich auch geeignet. Zusätzlich steht noch eine Augangsspannung von etwa 10 V ungeregelt zur Verfügung, z. B. um Relais, Lämpchen oder Lautsprecher-Treiberschaltungen zu versorgen (der EMUF darf an keinem seiner Anschlüsse diese Spannung erhalten!).

Es empfiehlt sich, alle peripheren Schaltungen, die direkt mit den I/O-Ports des EMUF verbunden sind, aus dem gleichen Netzteil zu versorgen, um zu vermeiden, daß sie Spannungen an die Ports liefern, während der EMUF noch keine Spannung erhält – dies könnte zu einer Beschädigung des 6532-Bausteins führen, da ein unzulässiger Ausgleichsstrom über die internen Schutzdioden fließt.



Einige wenige Bauelemente genügen, um den EMUF aus dem Netz mit Spannung zu versorgen. Statt des 7805 läßt sich ebensogut ein LM 309 als Regel-IC einsetzen

Wilhelm Springmann

Telefon-EMUF

Der Telefon-EMUF wird an das Telefon angeschlossen und wählt eine beliebige Telefonnummer, die eingegeben wird. Bei Bedarf kann die Rufnummer beliebig oft wiederholt werden. Außerdem kann der 6504-EMUF etwa 50 Rufnummern speichern und nach Eingabe einer Kennziffer wählen (das alles natürlich nur in Nebenstellenanlagen ohne Amtsberechtigung).

Die Bedienung des Telefon-EMUFs ist denkbar einfach: Nach Anschluß an das Telefon und Anlegen der Betriebsspannung leuchtet die grüne LED auf. Der Telefon-EMUF wartet nun auf die Eingabe der Kennziffer (1...255) des gewünschten Teilnehmers. Die eingetippten Ziffern werden dabei auf dem Display angezeigt. Die Eingabe wird mit der CR-Taste abgeschlossen. Sofort beginnt der Telefon-EMUF mit der Wahl der gewünschten Rufnummer, die gewählten Ziffern erscheinen auf dem Display.
Sollte der Anschluß besetzt sein, so genügt bei einem weiteren Versuch ein Druck auf die CR-Taste, um die Rufnummer erneut zu wählen.
Außer den fest gespeicherten Rufnummern ist es jedoch auch möglich, jede beliebige andere Nummer zu wählen. Nach Betätigen der „."-Taste leuchtet die gelbe LED auf und eine beliebige Rufnummer kann eingegeben werden, die nach Abschluß der Eingabe (CR) gewählt wird. Ein erneutes Drücken von CR wiederholt auch hier die zuletzt eingegebene Rufnummer.
Den Anschluß von Tastenfeld, LEDs und Relais zeigt *Bild 1*. Der Öffner des Relais wird mit einer Telefonader (a oder b) in Reihe geschaltet. Der Öffner stellt sicher, daß das Telefon auch bei ausgeschaltetem EMUF betriebsbereit ist.
Sinnvoll ist u. U. noch eine Reset-Taste, um Fehleingaben schnell löschen zu können. Eine Softwarelösung dieses Problems (Backspace-Taste) ist möglich, jedoch im vorliegenden Programm noch nicht vorhanden.

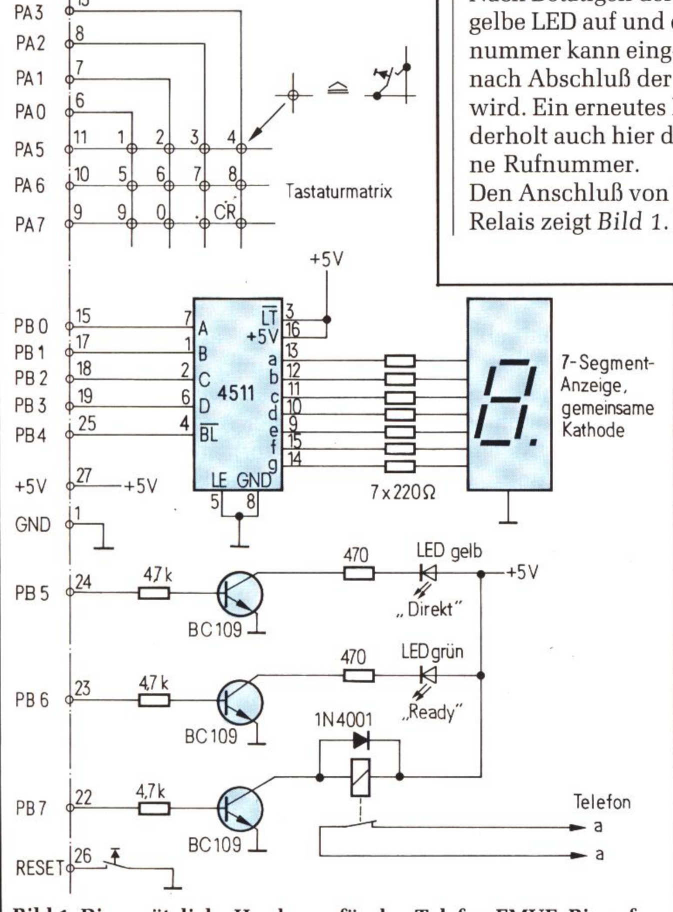


Bild 1. Die zusätzliche Hardware für den Telefon-EMUF. Bis auf die Tastatur läßt sich alles noch auf dem Lochrasterfeld des EMUFs unterbringen

Die Software zum Telefon-EMUF

Das Programm (*Bild 2*) wurde auf einem Apple-kompatiblen Computer entwikkelt, und zwar im Adreßbereich $4C00 $4FFF. Das Programm selbst belegt beim EMUF die Adressen $0C00...$0D7C. Ab $0D7D folgt die Tabelle der Rufnummern. Jede Rufnummer beginnt mit $FF als Delimiter, danach folgt eine Nummer mit beliebiger Stellenzahl. Nach der letzten Rufnummer folgt wieder ein $FF. Es empfiehlt sich, die freien Zellen des EPROMs mit $FF aufzufüllen, damit der Telefon-EMUF nach irrtümlicher Eingabe einer nichtdefinierten Rufnummer keinen Unsinn wählt.

Im Originalzustand lassen sich im EPROM noch ca. 50 zehnstellige Rufnummern unterbringen. Wem das nicht ausreicht, kann den EMUF leicht auf die volle Kapazität von 2 KByte ausbauen, wie im mc-EMUF-Sonderheft beschrieben. Das Programm muß dann allerdings an die geänderte Adreßlage angepaßt werden.

```
<0C00> 18 D8 A2 FF 8E 03 08 8E 01 08 9A A9 01 85 06 A9
<0C10> 41 8D 02 08 A9 7D 85 07 A9 0D 85 08 20 74 0C C9
<0C20> 0B F0 1E A5 02 D0 0B A5 06 D0 03 4C 49 0C A5 03
<0C30> 85 02 85 03 20 D8 0C A9 01 85 06 20 F6 0C 4C 0F
<0C40> 0C A9 20 8D 02 08 20 55 0C A0 00 84 05 84 06 20
<0C50> F6 0C 4C 0F 0C A0 00 84 05 20 9B 0C C9 0D F0 0C
<0C60> A4 05 99 09 00 20 5D 0D C8 4C 57 0C A4 05 A9 FF
<0C70> 99 09 00 60 A9 00 85 04 85 02 85 01 20 9B 0C C9
<0C80> 0D F0 17 C9 0B F0 13 85 01 20 5F 0D A5 02 0A 0A
<0C90> 65 02 0A 65 01 85 02 4C 7C 0C 60 20 B0 0C C9 FF
<0CA0> D0 05 85 04 4C 9B 0C A6 04 F0 F0 A2 00 86 04 60
<0CB0> A2 0F 8E 00 08 AD 00 08 09 F0 8D 00 08 A9 14 20
<0CC0> 54 0D AD 00 08 A2 0F DD 65 0D F0 05 CA 10 F8 30
<0CD0> 04 BD 71 0D 60 A9 FF 60 A9 00 AA A8 8D 02 08 B1
<0CE0> 07 C9 FF D0 05 E8 E4 02 F0 09 E6 07 D0 F1 E6 08
<0CF0> 4C DF 0C E6 07 60 84 05 A5 06 F0 05 B1 07 4C 04
<0D00> 0D B9 09 00 85 02 C9 FF F0 09 20 19 0D A4 05 C8
<0D10> 4C F6 0C A9 00 8D 02 08 60 C9 00 D0 02 A9 0A 85
<0D20> 00 A9 90 05 02 8D 02 08 A9 3A 20 54 0D A9 10 05
<0D30> 02 8D 02 08 C6 00 F0 08 A9 27 20 54 0D 4C 21 0D
<0D40> 20 49 0D A9 00 8D 02 08 60 A0 04 A9 F4 20 54 0D
<0D50> 88 D0 F8 60 8D 17 08 AD 16 08 D0 FB 60 09 30 09
<0D60> 10 8D 02 08 60 DE DD DB D7 BE BD BB B7 7E 7D 7B
<0D70> 77 01 02 03 04 05 06 07 08 09 00 0B 0D FF 00 08
<0D80> 09 05 01 01 07 03 05 04 FF 00 08 09 05 09 06 04
<0D90> 02 02 FF 00 08 09 05 09 08 04 02 03 FF 01 01 09
<0DA0> FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Bild 2. Die Software zum Telefon-EMUF. Zusätzlich ist als Reset-Vektor in 00 in 0FFC und 0C in 0FFD zu schreiben

Joachim Willmann

EMUF als Bordcomputer

Im Zeitalter steigender Benzinpreise, aber auch steigenden Umweltbewußtseins ist jeder Autofahrer bemüht, so benzinsparend wie möglich mit seinem Fahrzeug umzugehen. Eine wertvolle Hilfe dazu stellt das in diesem Beitrag vorgestellte Kraftstoffverbrauchsmeßgerät auf der Basis des 6504-EMUF dar.

Wie es sich für eine Lösung mit Mikroprozessor gehört, wurde Wert auf minimale Hardware, dafür aber auf maximalen Bedienungskomfort durch die Software gelegt. Neben der für ein solches Gerät obligatorischen Durchfluß- und Wegstreckenmessung kann dieser EMUF (siehe mc-EMUF-Sonderheft) auch noch als normale quarzgenaue Digitaluhr verwendet werden. Im Einzelnen sind folgende Anzeigemöglichkeiten vorhanden: Uhrzeit, Fahrzeit, gefahrene Wegstrecke, Gesamtverbrauch in Liter, Durchschnittsverbrauch in l/100 km, Durchschnittsgeschwindigkeit in km/h und Momentanverbrauch in l/100 km. Als Uhrzeitgeber wurde in der Schaltung das CMOS-IC M-755 (SGS) verwendet. Es handelt sich dabei um eine Uhr mit integrierter Anzeigensteuerung. Neben der Funktion als Uhr kann dieses IC auch über eine 6-Bit-Schnittstelle mit einer CPU verbunden werden und Daten, die in vier interne Register geschrieben werden, anzeigen. Über eine Interruptsteuerung wird die Uhr regelmäßig von der CPU abgefragt, um die Fahrzeit zu berechnen.

Bild 1. Der Hallgenerator SAS-241 von Siemens wird in den Tacho eingebaut

Die Durchflußmessung

Der Durchfluß, das heißt das Volumen des durch die Benzinleitung in den Vergaser fließenden Benzins, wird mit einem optischen Durchflußmesser der Firma KDM gemessen. Dieser gibt ziemlich genau 10 000 Impulse pro Liter Durchfluß ab. Der ebenfalls angebotene induktive Durchflußmesser hat sich in der Praxis nicht so bewährt, da zum einen das schwache Ausgangssignal der Induktionsspule durch die Zündung gestört werden kann, und zum andern eventuell vorhandene kleine Metallspäne aus dem Tank an den Magneten des Durchlußmessers haften bleiben und diesen zum Klemmen bringen. Das Gerät eignet sich für alle Motoren ohne Rücklaufleitung vom Vergaser in den Tank. Nicht geeignet ist es dagegen für die meisten Einspritzmotoren.

Die Wegstreckenmessung

Die Wegstrecke wird über einen im Tachometer angebrachten Wegstreckengeber gemessen. Im Tachometer rotiert ein mit der Tachowelle verbundener Magnet. Bringt man nun in die Nähe dieses Magneten (der optimale Punkt muß durch Versuche ermittelt werden) ein magnetfeldempfindliches Bauteil, wie in diesem Fall ein Hall-Generator (*Bild 1*), so erhält man durch Zählung der abgegebenen Impulse ein exaktes Maß für die gefahrene Wegstrecke. Hierzu muß man nur noch die Anzahl der Impulse bzw. der Umdrehungen des Magneten für eine feste Wegstrecke ermitteln. Da diese Anzahl von Fahrzeug zu Fahrzeug verschieden ist, muß sie zunächst entweder von Hand oder mit dem später gezeigten Abgleichprogramm ermittelt werden.

Die Auswertung

Aus den drei Eingangsgrößen Zeit, Durchfluß und Wegstrecke werden dann durch das Programm die Ausgabegrößen berechnet. Der Rechenvorgang findet nach jedem Meßzyklus statt. Ein Meßzyklus ist beendet, wenn das Fahrzeug 100 Meter weiter gefahren ist. Die Durchschnittswerte für Verbrauch und Geschwindigkeit werden nach jedem ganzen Kilometer berechnet.

Die Bedienung des EMUF ist denkbar einfach. Wird eine Taste gedrückt, so leuchtet danach die zugehörige LED auf. Damit ist auch klar, welcher Meßwert sich gerade in der Anzeige befindet. *Bild 2* zeigt die vorhandenen Tasten und deren Funktion. Drückt man die Tasten 0, 1, 6 und 7 gleichzeitig, so kommt man

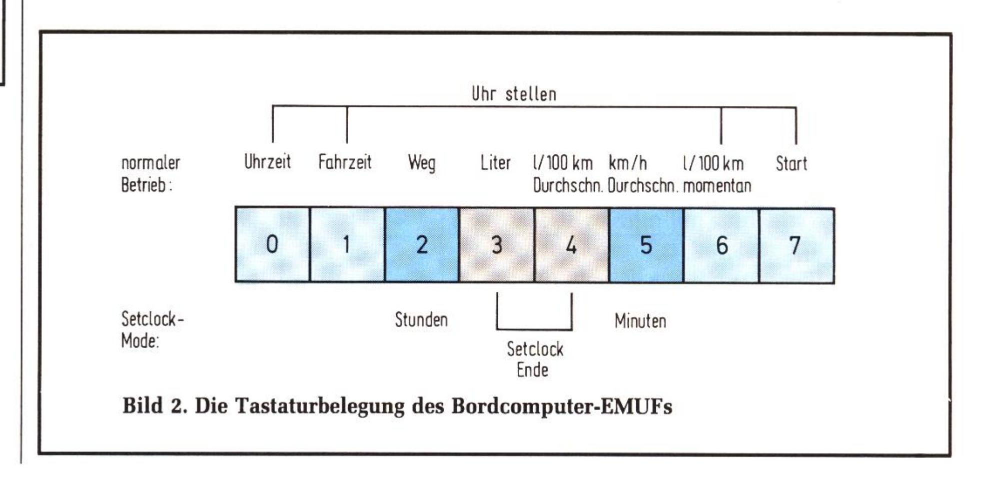
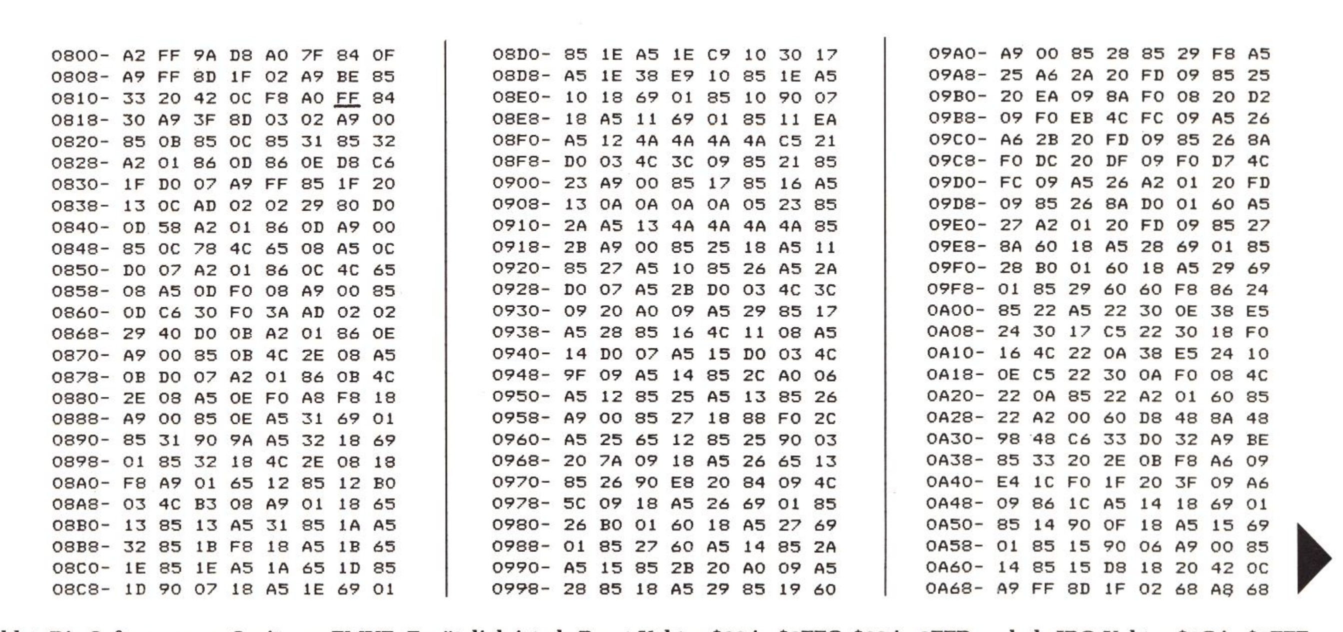


Bild 2. Die Tastaturbelegung des Bordcomputer-EMUFs

Bild 3. Die zusätzliche Hardware für den Spritspar-EMUF. Die Versorgungsspannung sollte direkt von der Autobatterie, also nicht über das Zündschloß kommen. Den Drehkondensator zwischen Pin2 des M-755 und Masse stellt man so ein, daß der Oszillator sicher anschwingt



```
0800- A2 FF 9A D8 A0 7F 84 0F
0808- A9 FF 8D 1F 02 A9 BE 85
0810- 33 20 42 0C F8 A0 FF 84
0818- 30 A9 3F 8D 03 02 A9 00
0820- 85 0B 85 0C 85 31 85 32
0828- A2 01 86 0D 86 0E D8 C6
0830- 1F D0 07 A9 FF 85 1F 20
0838- 13 0C AD 02 02 29 80 D0
0840- 0D 58 A2 01 86 0D A9 00
0848- 85 0C 78 4C 65 08 A5 0C
0850- D0 07 A2 01 86 0C 4C 65
0858- 08 A5 0D F0 08 A9 00 85
0860- 0D C6 30 F0 3A AD 02 02
0868- 29 40 D0 0B A2 01 86 0E
0870- A9 00 85 0B 4C 2E 08 A5
0878- 0B D0 07 A2 01 86 0B 4C
0880- 2E 08 A5 0E F0 A8 F8 18
0888- A9 00 85 0E A5 31 69 01
0890- 85 31 90 9A A5 32 18 69
0898- 01 85 32 18 4C 2E 08 18
08A0- F8 A9 01 65 12 85 12 B0
08A8- 03 4C B3 08 A9 01 18 65
08B0- 13 85 13 A5 31 85 1A A5
08B8- 32 85 1B F8 18 A5 1B 65
08C0- 1E 85 1E A5 1A 65 1D 85
08C8- 1D 90 07 18 A5 1E 69 01

08D0- 85 1E A5 1E C9 10 30 17
08D8- A5 1E 38 E9 10 85 1E A5
08E0- 10 18 69 01 85 10 90 07
08E8- 18 A5 11 69 01 85 11 EA
08F0- A5 12 4A 4A 4A 4A C5 21
08F8- D0 03 4C 3C 09 85 21 85
0900- 23 A9 00 85 17 85 16 A5
0908- 13 0A 0A 0A 0A 05 23 85
0910- 2A A5 13 4A 4A 4A 4A 85
0918- 2B A9 00 85 25 18 A5 11
0920- 85 27 A5 10 85 26 A5 2A
0928- D0 07 A5 2B D0 03 4C 3C
0930- 09 20 A0 09 A5 29 85 17
0938- A5 28 85 16 4C 11 08 A5
0940- 14 D0 07 A5 15 D0 03 4C
0948- 9F 09 A5 14 85 2C A0 06
0950- A5 12 85 25 A5 13 85 26
0958- A9 00 85 27 18 88 F0 2C
0960- A5 25 65 12 85 25 90 03
0968- 20 7A 09 18 A5 26 65 13
0970- 85 26 90 E8 20 84 09 4C
0978- 5C 09 18 A5 26 69 01 85
0980- 26 B0 01 60 18 A5 27 69
0988- 01 85 27 60 A5 14 85 2A
0990- A5 15 85 2B 20 A0 09 A5
0998- 28 85 18 A5 29 85 19 60

09A0- A9 00 85 28 85 29 F8 A5
09A8- 25 A6 2A 20 FD 09 85 25
09B0- 20 EA 09 8A F0 08 20 D2
09B8- 09 F0 EB 4C FC 09 A5 26
09C0- A6 2B 20 FD 09 85 26 8A
09C8- F0 DC 20 DF 09 F0 D7 4C
09D0- FC 09 A5 26 A2 01 20 FD
09D8- 09 85 26 8A D0 01 60 A5
09E0- 27 A2 01 20 FD 09 85 27
09E8- 8A 60 18 A5 28 69 01 85
09F0- 28 B0 01 60 18 A5 29 69
09F8- 01 85 29 60 60 F8 86 24
0A00- 85 22 A5 22 30 0E 38 E5
0A08- 24 30 17 C5 22 30 18 F0
0A10- 16 4C 22 0A 38 E5 24 10
0A18- 0E C5 22 30 0A F0 08 4C
0A20- 22 0A 85 22 A2 01 60 85
0A28- 22 A2 00 60 D8 48 8A 48
0A30- 98 48 C6 33 D0 32 A9 BE
0A38- 85 33 20 2E 0B F8 A6 09
0A40- E4 1C F0 1F 20 3F 09 A6
0A48- 09 86 1C A5 14 18 69 01
0A50- 85 14 90 0F 18 A5 15 69
0A58- 01 85 15 90 06 A9 00 85
0A60- 14 85 15 D8 18 20 42 0C
0A68- A9 FF 8D 1F 02 68 A8 68
```

Bild 4. Die Software zum Spritspar-EMUF. Zusätzlich ist als Reset-Vektor $00 in $0FFC, $08 in 0FFD und als IRQ-Vektor $2C in $0FFE und $0A in $0FFF zu programmieren. Die unterstrichene Speicherstelle muß vor dem Programmieren mit dem Abgleichwert belegt werden

in den Setclock-Modus. In diesem Modus kann die Uhr gestellt werden. Das Stellen wird durch das gleichzeitige Drücken von Taste 3 und 4 beendet. Wird der EMUF mit Schalter S 8 ausgeschaltet, so wird die CMOS-Uhr gleichzeitig auf Backup-Betrieb umgestellt und über den Spannungsteiler R1/R2 aus der Autobatterie versorgt. Aus diesem Grund sollte das Gerät auch nicht über das Zündschloß, sondern direkt mit 12 V betrieben werden. Der Stromverbrauch beträgt dann nur noch einige µA.

Aufbau und Abgleich

Beim Aufbau gibt es grundsätzlich zwei Möglichkeiten. Entweder man besorgt sich einen EMUF-Bausatz und baut die Schaltung in *Bild 3* bis auf Tastatur und Anzeigen auf dem zusätzlichen Lochrasterfeld auf, oder man baut den kompletten EMUF mit der Schaltung von Bild 3 zusammen auf einer eigenen Lochrasterplatte auf. Die zweite Möglichkeit hat den Vorteil, daß das Gerät kompakter aufgebaut und den Einbaugegebenheiten im Auto angepaßt werden kann.
Um das EPROM zu programmieren, wird nun noch die Zahl der Impulse pro 100 Meter des Tachogebers benötigt. Dies ist gleichzeitig auch der einzige Abgleich- bzw. Anpassungspunkt. Diese Zahl kann ermittelt werden, indem man den ausgebauten Tacho von Hand dreht und die Ausschläge eines an den Geber angeschlossenen Meßgerätes zählt. Die Anzahl dieser Ausschläge für eine Wegstrecke von 100 m wird nun in eine Hexadezimalzahl umgewandelt und in die im Listing in *Bild 4* unterstrichene EPROM-Speicherstelle geschrieben. Da diese Möglichkeit im allgemeinen recht langwierig und unter Umständen auch ungenau ist, kann man sich auch ein EPROM mit dem in *Bild 5* angegebenen Abgleichprogramm programmieren. Setzt man dieses in den fertig aufgebauten EMUF ein, so wird auf dem Display die Anzahl der ankommenden Impulse angezeigt. Durch Drücken der Start-Taste kann diese auf Null rückgesetzt werden. Man fährt nun eine Wegstrecke von z. B. 1 km oder 10 km und teilt die ermittelte Impulszahl durch 10 bzw. 100. Als Hexadezimalzahl fügt man diese Zahl dann genauso wie bei der ersten Möglichkeit in das Listing von Bild 3 ein. Das Hauptprogrmm (Bild 4) sowie das Abgleichprogramm (Bild 5) wurden auf einem Apple-kompatiblen Computer entwickelt. Das Hauptprogramm belegt beim EMUF die Adressen \$0800...\$0D2F, was die im EMUF-Sonderheft beschriebene erweiterte Adressierung notwendig macht. Hat man das Programm einschließlich der Abgleichspeicherstelle in ein 2716-EPROM programmiert und die beiden Geber in den PKW eingebaut, so kann der Spritspar-EMUF ohne weitere Abgleichmaßnahmen in Betrieb genommen werden.

```
0800- A2 FF 9A D8 A9 FF 85 0A
0808- A9 00 85 08 85 09 85 01
0810- 85 02 20 58 08 A9 3F 8D
0818- 03 02 AD 02 02 29 80 D0
0820- 0B A2 01 86 07 A9 00 85
0828- 06 4C 1A 08 A5 06 D0 07
0830- A2 01 86 06 4C 1A 08 A5
0838- 07 F0 DF A9 00 85 07 20
0840- E7 08 F8 18 A9 01 65 08
0848- 85 08 90 CE 18 A9 01 65
0850- 09 85 09 90 C5 4C 04 08
0858- D8 A9 3F 8D 03 02 A2 00
0860- BD 20 09 8D 02 02 E8 E0
0868- 06 D0 F5 A9 0E 85 03 A5
0870- 01 85 04 20 D1 08 20 CA
0878- 08 20 A5 08 C6 03 20 D1
0880- 08 20 C1 08 20 A5 08 A5
0888- 02 C9 10 10 0A 18 69 F0
0890- C9 F0 D0 03 18 69 0F 85
0898- 04 C6 03 A5 03 C9 0A F0
08A0- 03 4C 73 08 60 A5 05 49
08A8- 0F 09 30 8D 02 02 A5 05
08B0- 49 0F 09 20 8D 02 02 A5
08B8- 05 49 0F 09 30 8D 02 02
08C0- 60 A5 04 4A 4A 4A 4A 85
08C8- 05 60 A5 04 29 0F 85 05
08D0- 60 A5 03 09 30 8D 02 02
08D8- A5 03 09 10 8D 02 02 A5
08E0- 03 09 30 8D 02 02 60 D8
08E8- A9 00 8D 01 02 AD 00 02
08F0- C9 FF D0 03 4C FC 08 AD
08F8- 00 02 85 0A 20 00 09 60
0900- A5 0A C9 7F D0 0E A0 09
0908- A9 00 99 00 00 88 D0 FA
0910- A0 FF 84 0A A5 08 85 01
0918- A5 09 85 02 20 58 08 60
0920- 3F 1F 3F 3F 2F 3F FF FF
0928- FF FF FF FF FF FF FF FF
```

Bild 5. Das Abgleichprogramm. Es dient dazu, das Hauptprogramm an das jeweilige Fahrzeug anzupassen. Zusätzlich ist als Reset-Vektor \$00 in \$0FFC und \$08 in \$0FFD zu programmieren

```
0A70- AA 68 40 A9 0B 85 0A 20
0A78- 06 0B A9 3F 8D 03 02 A2
0A80- 00 BD 0A 0D 8D 02 02 E8
0A88- E0.0D D0 F5 60 A9 3F 8D
0A90- 03 02 85 2F A0 00 84 2D
0A98- 84 2E 20 EC 0A A0 FF 84
0AA0- 0F 20 13 0C F8 A5 0F C9
0AA8- FB D0 10 A5 2E 18 69 01
0AB0- C9 25 D0 02 A9 00 85 2E
0AB8- 20 EC 0A A5 0F C9 DF D0
0AC0- 10 A5 2D 18 69 01 C9 60
0AC8- D0 02 A9 00 85 2D 20 EC
0AD0- 0A A5 0F C9 E7 D0 C6 A2
0AD8- 00 86 2F BD 17 0D 8D 02
0AE0- 02 E8 E0 06 D0 F5 A0 10
0AE8- 84 0F D8 60 A5 2D 85 01
0AF0- A5 2E 85 02 20 80 0B 20
0AF8- FB 0A 60 A0 FF A2 FF CA
0B00- D0 FD 88 D0 F8 60 A9 3F
0B08- 8D 03 02 A2 00 BD 1D 0D
0B10- 8D 02 02 E8 E0 09 D0 F5
0B18- A5 0A 09 30 8D 02 02 A5
0B20- 0A 09 20 8D 02 02 A5 0A
0B28- 09 30 8D 02 02 60 A2 00
0B30- A0 3F 8C 03 02 BD 26 0D
0B38- 8D 02 02 E8 E0 02 D0 F5
0B40- 20 6E 0B 8C 03 02 85 08
0B48- BD 26 0D 8D 02 02 E8 E0
0B50- 0A D0 F5 20 6E 0B 85 09
0B58- 8C 03 02 BD 26 0D 8D 02
0B60- 02 E8 E0 0D D0 F5 A5 08
0B68- 85 05 20 D1 0B 60 A9 30
0B70- 8D 03 02 A9 00 8D 02 02
0B78- AD 02 02 49 0F 29 0F 60
0B80- D8 A9 3F 8D 03 02 A2 00
0B88- BD 32 0D 8D 02 02 E8 E0
0B90- 06 D0 F5 A9 0E 85 03 A5
0B98- 01 85 04 20 FD 0B 20 F6
0BA0- 0B 20 D1 0B C6 03 20 FD
0BA8- 0B 20 ED 0B 20 D1 0B A5
0BB0- 02 A4 2F D0 0E C9 10 10
0BB8- 0A 18 69 F0 C9 F0 D0 03
0BC0- 18 69 0F 85 04 C6 03 A5
0BC8- 03 C9 0A F0 03 4C 9B 0B
0BD0- 60 A5 05 49 0F 09 30 8D
0BD8- 02 02 A5 05 49 0F 09 20
0BE0- 8D 02 02 A5 05 49 0F 09
0BE8- 30 8D 02 02 60 A5 04 4A
0BF0- 4A 4A 4A 85 05 60 A5 04
0BF8- 29 0F 85 05 60 A5 03 09
0C00- 30 8D 02 02 A5 03 09 10
0C08- 8D 02 02 A5 03 09 30 8D
0C10- 02 02 60 D8 A9 00 8D 01
0C18- 02 AD 00 02 C9 FF D0 0B
0C20- A9 FF 8D 01 02 A5 0F 8D
0C28- 00 02 60 AD 00 02 85 0F
0C30- 48 A9 FF 8D 01 02 68 8D
0C38- 00 02 A5 2F D0 03 20 42
0C40- 0C 60 A5 0F C9 FE D0 03
0C48- 20 73 0A A5 0F C9 FD D0
0C50- 12 A5 14 85 01 A5 15 85
0C58- 02 A9 0F 85 0A 20 06 0B
0C60- 20 80 0B A5 0F C9 FB D0
0C68- 12 A5 12 85 01 A5 13 85
0C70- 02 A9 0D 85 0A 20 06 0B
0C78- 20 80 0B A5 0F C9 F7 D0
0C80- 12 A5 10 85 01 A5 11 85
0C88- 02 A9 0D 85 0A 20 06 0B
0C90- 20 80 0B A5 0F C9 EF D0
0C98- 12 A5 16 85 01 A5 17 85
0CA0- 02 A9 0D 85 0A 20 06 0B
0CA8- 20 80 0B A5 0F C9 DF D0
0CB0- 12 A5 18 85 01 A5 19 85
0CB8- 02 A9 0F 85 0A 20 06 0B
0CC0- 20 80 0B A5 0F C9 BF D0
0CC8- 12 A5 1A 85 01 A5 1B 85
0CD0- 02 A9 0D 85 0A 20 06 0B
0CD8- 20 80 0B A5 0F C9 7F D0
0CE0- 12 D8 A0 32 A9 00 99 00
0CE8- 00 88 D0 FA A0 FE 84 0F
0CF0- 4C 42 0C A5 0F C9 3C D0
0CF8- 06 20 8D 0A 20 73 0A A9
0D00- FF 8D 01 02 A5 0F 8D 00
0D08- 02 60 3F 1F 3F 36 26 36
0D10- 3A 1A 3A 3B 2B 3B 3F 3F
0D18- 1F 3F 34 24 34 3F 1F 3F
0D20- 3F 2F 3F 3A 1A 3A 3F 1F
0D28- 3F 1F 3F 36 26 36 3E 1E
0D30- 3F 1F 3F 1F 3F 3F 2F 3F
0D38- FF FF FF FF FF FF FF FF
0D40- FF FF FF FF FF FF FF FF
```

Alfred Schön

Centronics-Interface

EMUF am seriellen Commodore-Bus

Bei Centronics-Schnittstellen für den C-64 überwiegen die Software-Lösungen, die den User-Port praktisch vollständig belegen, das zugehörige Programm muß erst von der Diskette geladen werden. Das EMUF-Interface hingegen ist nach dem Einschalten sofort betriebsbereit und läßt den User-Port frei. Durch den Anschluß am seriellen Bus ist das Interface gleichermaßen für den VC-20, C-64 oder C-128 (im C-64-Modus) geeignet.

Der Betrieb eines Druckers mit Centronics-Schnittstelle an einem der genannten Commodore-Rechner setzt voraus, daß im wesentlichen Texte gedruckt werden sollen. Die vielen Grafik-Sonderzeichen der Commodore-Rechner lassen sich zwar auch auf einem Drucker eines anderen Herstellers darstellen, aber zum einen muß dieser Drucker dann grafikfähig sein (die Software wird dadurch stark vom Drucker abhängig), zum anderen werden die meisten Grafikzeichen sehr selten benötigt. Zur Darstellung der schon etwas häufiger vorkommenden Bildschirm-Steuerzeichen hingegen kann man auf entsprechende mnemonische Begriffe ausweichen.

Umcodieren mit Tabelle

Der etwas absonderliche Zeichensatz der Commodore-Rechner muß an die Fähigkeiten von normalen Druckern angepaßt werden, insbesondere sind Korrekturen bezüglich der Groß-/Kleinschreibung erforderlich. Zu diesem Zweck enthält das EPROM des EMUFs eine 256 Byte große Tabelle, in der diese Umcodierung ohne große Schwierigkeiten vorgenommen werden kann. *Bild 1* zeigt das vollständige Programm, das in ein EPROM 2716 übertragen und in den EMUF eingesetzt werden muß. Die Routinen zur Abwicklung der Datenübertragung zwischen EMUF und Commodore-Rechner sind an [1] angelehnt und nur geringfügig modifiziert. Die Tabelle zur Umcodierung beginnt bei $E00 und endet bei $EFF. Der Wert des über den seriellen Bus ankommenden Zeichens wird als Index verwendet, unter dem man dann das zu sendende Zeichen in der Tabelle findet. So wird beispielsweise aus dem A des C-64 ($41) auf dem Drucker ein a ($61, in der Tabelle zu finden bei $E41). Daraus wird auch gleich ersichtlich, daß Alphazeichen ohne Shift in Kleinschreibung dargestellt werden, mit Shift hingegen als Großbuchstaben.
Die vielen Nullen in der Tabelle stehen da, wo die Zeichen des Rechners von einem Drucker mit Standard-ASCII-Zeichensatz nicht dargestellt werden können. Die Null für nicht druckbare Zeichen wurde deshalb gewählt, weil die Centronics-Ausgaberoutine dieses Zeichen nicht sendet, sondern es gleich unter den Tisch fallen läßt.

Mnemonics für Steuerzeichen

Die Drucker-Ausgaberoutine ist in *Bild 2* dokumentiert. Sie unterscheidet zwischen Zeichen, die direkt gedruckt werden oder in eine Zeichenkette umgewandelt werden sollen. Zeichen mit einem Wert <$80 werden direkt gedruckt, diejenigen ab $80 aufwärts in einen String umgewandelt. Dieser String ist dem Zeichen zugeordnet in einer zweiten Tabelle, die bei $F00 beginnt. Nach Löschen des höchstwertigen Bits des Tabellenwertes wird der Rest wiederum als Index für die zweite Tabelle benutzt. Zuvor

```
0c00  78 a2 ff 9a d8 20 36 0c 20 68 0c 20 5e 0c ad 00  05b8
0c10  08 29 20 d0 f6 20 76 0c 20 6e 0c 20 a1 0c 20 87  04c7
0c20  0c b0 dd 20 a1 0c 24 02 f0 06 a5 01 c9 3f f0 d0  06f0
0c30  20 f6 0c 4c 23 0c a9 02 8d 01 08 ad 00 08 09 02  039e
0c40  8d 00 08 a9 ff 8d 03 08 a9 00 8d 02 08 a9 00 85  0543
0c50  03 a9 0e 85 04 a9 00 85 05 a9 0f 85 06 60 ad 00  04c6
0c60  08 30 fb 29 20 d0 f7 60 ad 00 08 10 fb 60 ad 00  0670
0c70  08 29 20 f0 f9 60 ad 00 08 29 1f 8d 00 08 ad 01  04da
0c80  08 09 40 8d 01 08 60 a5 01 c9 24 18 f0 01 38 60  047b
0c90  ad 00 08 29 20 d0 f9 60 ad 01 08 29 bf 8d 01 08  055b
0ca0  60 20 6e 0c 20 98 0c a9 0f 85 00 ad 00 08 29 20  03f9
0cb0  f0 0f c6 00 d0 f5 20 76 0c a2 0c ca d0 fd 20 98  0829
0cc0  0c a2 08 20 90 0c 20 6e 0c ad 00 08 29 40 0a 0a  033e
0cd0  66 01 ca d0 ee ad 00 08 30 04 a9 ff d0 02 a9 00  06fb
0ce0  85 02 20 90 0c ad 00 08 29 1f 8d 00 08 ad 01 08  038b
0cf0  09 40 8d 01 08 60 a5 02 d0 16 a5 01 c9 0d d0 0d  0525
0d00  48 ad 00 08 4a 90 05 a9 0a 20 2f 0d 68 20 11 0d  0391
0d10  60 a8 b1 03 f0 26 10 17 29 7f 0a ea a8 a9 5b 20  0661
0d20  2f 0d a2 02 b1 05 20 2f 0d c8 ca d0 f7 a9 5d 8d  06de
0d30  02 08 20 3d 0d ad 00 08 29 04 d0 f9 60 ad 00 08  0434
0d40  29 fd 8d 00 08 09 02 8d 00 08 60 ff ff ff ff ff  07b6
0e00  00 01 02 03 04 80 06 07 08 09 0a 0b 0c 0d 0e 0f  00f3
0e10  10 81 82 83 14 15 16 17 18 19 1a 1b 84 85 86 87  0468
0e20  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f  0278
0e30  30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f  0378
0e40  40 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f  0658
0e50  70 71 72 73 74 75 76 77 78 79 7a 5b 7c 5d 5e 7f  0718
0e60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
0e70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
0e80  00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f  0078
0e90  88 89 8a 8b 14 15 16 17 18 19 1a 1b 8c 8d 8e 8f  0518
0ea0  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0020
0eb0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
0ec0  60 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f  0498
0ed0  50 51 52 53 54 55 56 57 58 59 5a 7b 5c 7d 7e 5f  05d8
0ee0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
0ef0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
0f00  77 74 63 64 72 6e 68 6f 72 64 63 72 67 6e 62 6c  06b7
0f10  62 6b 63 75 72 66 63 73 70 75 63 6c 79 77 63 79  06d3
0ff0  ff ff ff ff ff ff ff ff ff ff ff ff 00 0c 00 0c  0c0c
```

Bild 1. Das Schnittstellen-Programm als Hexdump mit Prüfsumme pro Zeile. Die Adresse $0C00 entspricht im EPROM der Adresse $0

```
0171  0D11                ;AUSGABEROUTINEN FUER DEN DRUCKER
0172  0D11  A8          OUTALL TAY                    ;INDEX FUER TAB1
0173  0D12  B1 03              LDA (CONTAB),Y ;ZEICHEN HOLEN
0174  0D14  F0 26              BEQ ENDOUT ;NULL NICHT SENDEN
0175  0D16  10 17              BPL OUT    ;KEIN STRING
0176  0D18  29 7F       STRING AND #%01111111   ;BIT 7 LOESCHEN
0177  0D1A  0A                 ASL A      ;FAKTOR 2
0178  0D1B  EA                 NOP        ;RESERVE F. FAKTOR 4
0179  0D1C  A8                 TAY        ;INDEX FUER TAB2
0180  0D1D  A9 5B       STRON  LDA #'['
0181  0D1F  20 2F 0D           JSR OUT
0182  0D22  A2 02              LDX #2     ;STRINGLAENGE = 2
0183  0D24  B1 05       STRCHR LDA (CTRTAB),Y   ;STRINGZEICHEN
0184  0D26  20 2F 0D           JSR OUT
0185  0D29  C8                 INY
0186  0D2A  CA                 DEX
0187  0D2B  D0 F7              BNE STRCHR ;WEITER MIT STRING
0188  0D2D  A9 5D       STROUT LDA #']'
0189  0D2F  8D 02 08    OUT    STA PRTB         ;ZEICHEN LIEGT AN
0190  0D32  20 3D 0D           JSR STROBE ;STROBE
0191  0D35  AD 00 08    WAIT   LDA PRTA          ;WARTEN AUF BUSY=LOW
0192  0D38  29 04              AND #%00000100
0193  0D3A  D0 F9              BNE WAIT
0194  0D3C  60          ENDOUT RTS
0195  0D3D              ;
0196  0D3D  AD 00 08    STROBE LDA PRTA
0197  0D40  29 FD              AND #%11111101 ;STROBE=LOW
0198  0D42  8D 00 08           STA PRTA
0199  0D45  09 02              ORA #%00000010 ;STROBE=HIGH
0200  0D47  8D 00 08           STA PRTA
0201  0D4A  60                 RTS
```

Bild 2. Die Centronics-Ausgaberoutine als Source-Listing

muß der Tabelle-Offset allerdings der Stringlänge entsprechend umgerechnet werden (hier mit ASL A, wodurch sich Längen von 2, 4 oder 8 anbieten).

In der vorliegenden Form sind die Strings jeweils zwei Zeichen lang, die vom Drucker in eckigen Klammern dargestellt werden. Aus dem Steuerzeichen für HOME wird dann auf dem Drucker [ho] oder [cl] für Cursor nach links. Nun sind zwei Zeichen als Abkürzung dem einen oder anderen zu wenig. Im Programm ist deshalb die Erweiterung auf vier Zeichen (ohne Klammern) vorgesehen. Dazu ist das NOP in Speicherstelle $D1B (Opcode $EA) durch ein zweites ASL (Opcode $0A) zu ersetzen, das X-Register wird bei der Ausgabe statt mit 2 mit 4 geladen (Adresse $D23). In die Tabelle sind die Zeichenfolgen mit je vier Zeichen einzutragen. HOME beispielsweise könnte dann direkt hingeschrieben werden.

Da im vorliegenden Programm erst 16 Zeichen in Strings umgewandelt werden, kann die Stringtabelle bei Bedarf noch erheblich erweitert werden. Belegt sind zunächst nur die Codierungen $80 für die Farbe Weiß ($E05 in der Tabelle) bis $8F (Cyan, $E9F in der Tabelle).

Das Schnittstellen-Handling

Die einfachste Art der Centronics-Schnittstelle besteht aus den acht Daten-Leitungen sowie Strobe und Busy zum Abwickeln der Datenübertragung. Eine Zeitüberwachung ist im Programm nicht eingebaut, so daß sich das Interface einschließlich angeschlossenem Rechner in einer Warteschleife befindet, falls das Busy-Signal nicht auf Low geht.

Der Anschluß des EMUFs an den Computer sowie an den Drucker zeigt *Bild 3*. Die Buffer 7407 zwischen EMUF und Centronics-Stecker sind deshalb eingefügt, weil bei manchen Druckern mit Centronics-Schnittstelle die Eingänge mit Pull-Up-Widerständen von 1 kΩ oder kleiner beschaltet sind, so daß der RIOT-Baustein 6532 des EMUFs u. U. etwas überfordert ist; vor allem dann, wenn der EMUF aus-, der Drucker hingegen noch eingeschaltet ist. Die Buffer zwischenzuschalten ist also in jedem Fall beruhigend, aber nicht immer notwendig. Im Zweifelsfall sollte das Handbuch des Druckerherstellers zu Rate gezogen werden, in dem (hoffentlich) die Hardware-Konfiguration der Centronics-Schnittstelle zu finden ist. Falls nicht, kann man eine Datenleitung des Druckers auf Masse legen, den dabei fließenden Strom messen und somit einen Anhaltspunkt für die Belastung des Portbausteins bekommen.

Literatur

[1] *Ruepp, A.*: EMUF als serielles Interface. mc 1985, Heft 11, Seite 132.

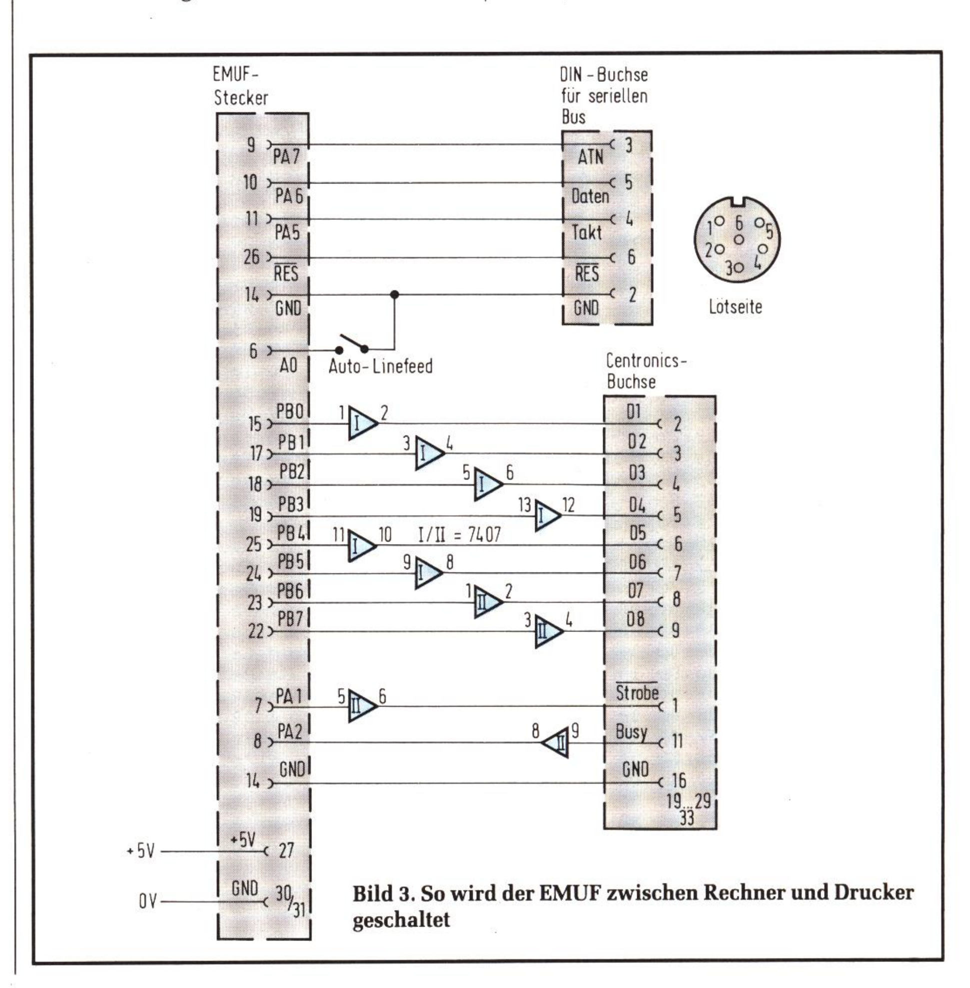


Bild 3. So wird der EMUF zwischen Rechner und Drucker geschaltet

Stephan Thienel

EMUF mal zwei

So wie ein „Mädchen für alles“ manchmal nicht genügend Hände hat, so fehlten dem 6504-EMUF (mc 1981, Heft 2, und EMUF-Sonderheft) für manche Anwendungen noch Füße, sprich I/O-Leitungen. Hier wird ein neuer, erweiterter 6504-EMUF vorgestellt, der in jeder Beziehung doppelt so viel kann wie der alte: 2 KByte EPROM-Bereich, 256 Byte RAM, 32 I/O-Leitungen und zwei Timer. Der Preis für einen Bausatz bleibt trotzdem unter 100 DM.

Die in Heft 2/1981 und im EMUF-Sonderheft veröffentlichte Schaltung wurde mit geringfügigen Erweiterungen übernommen (*Bild 1*). Zur Adressierung des zweiten 1-KByte-Bereiches des EPROMs 2716 dient die bisher ungenutzte Leitung A12 des 6504. Sie wurde mit A10 des 2716 verbunden. Dieser Adressierungskniff wurde aus Gründen der Schaltungsvereinfachung angewendet, auch wenn dadurch der EPROM-Bereich in die zwei Teile $0C00...$0FFF und $1C00...$1FFF zerfällt. Dies stellt keinen echten Nachteil dar, nachdem jeder gute Assembler in der Lage ist, darauf Rücksicht zu nehmen. Für handgeschriebene Programme bietet sich die „Sandwich-Methode“ an, d. h., Unterprogramme und Tabellen werden vom EPROM-Ende her beschrieben, das Hauptprogramm vom EPROM-Anfang her.

Ein zweiter I/O-Baustein vom Typ 6532 mit einem weiteren Timer und zusätzlichen 128 Byte RAM verdoppelt auch die Zahl der Ports. Er ist zum ersten parallelgeschaltet, nur die CS1-Leitung liegt an A7 statt an +5 V. Auch die CS1-Leitung des ersten 6532 wurde von +5 V getrennt und über einen Inverter ebenfalls mit A7 verbunden. Die *Tabelle 1* zeigt die Verteilung der Adreßbereiche. Sollen Programme des alten EMUF verwendet werden, so müssen lediglich die Vektoren für Reset und IRQ an das Ende des EPROMs gelegt werden. Es ist möglich, nur einen 6532 zu bestücken, doch muß dann eventuell dafür gesorgt werden, daß der Stack richtig initialisiert wird (LDX #$7F, TXS...).

Etwa zwei Drittel der Platinenfläche sind durch die EMUF-Schaltung selbst belegt. Der Rest besteht aus einem Lochraster mit 33 × 16 Löt-Punkten. Auf seine innerste Reihe ist der Port A gelegt. Dies erleichtert die Verbindung eventueller Zusatzschaltungen mit dem EMUF. Für den Kontakt mit der Außenwelt sind acht weitere Rasterpunkte auf die 64polige Steckerleiste herausgeführt. *Bild 2* zeigt die Belegung der Steckerleiste und der innersten Lochraster-Reihe.

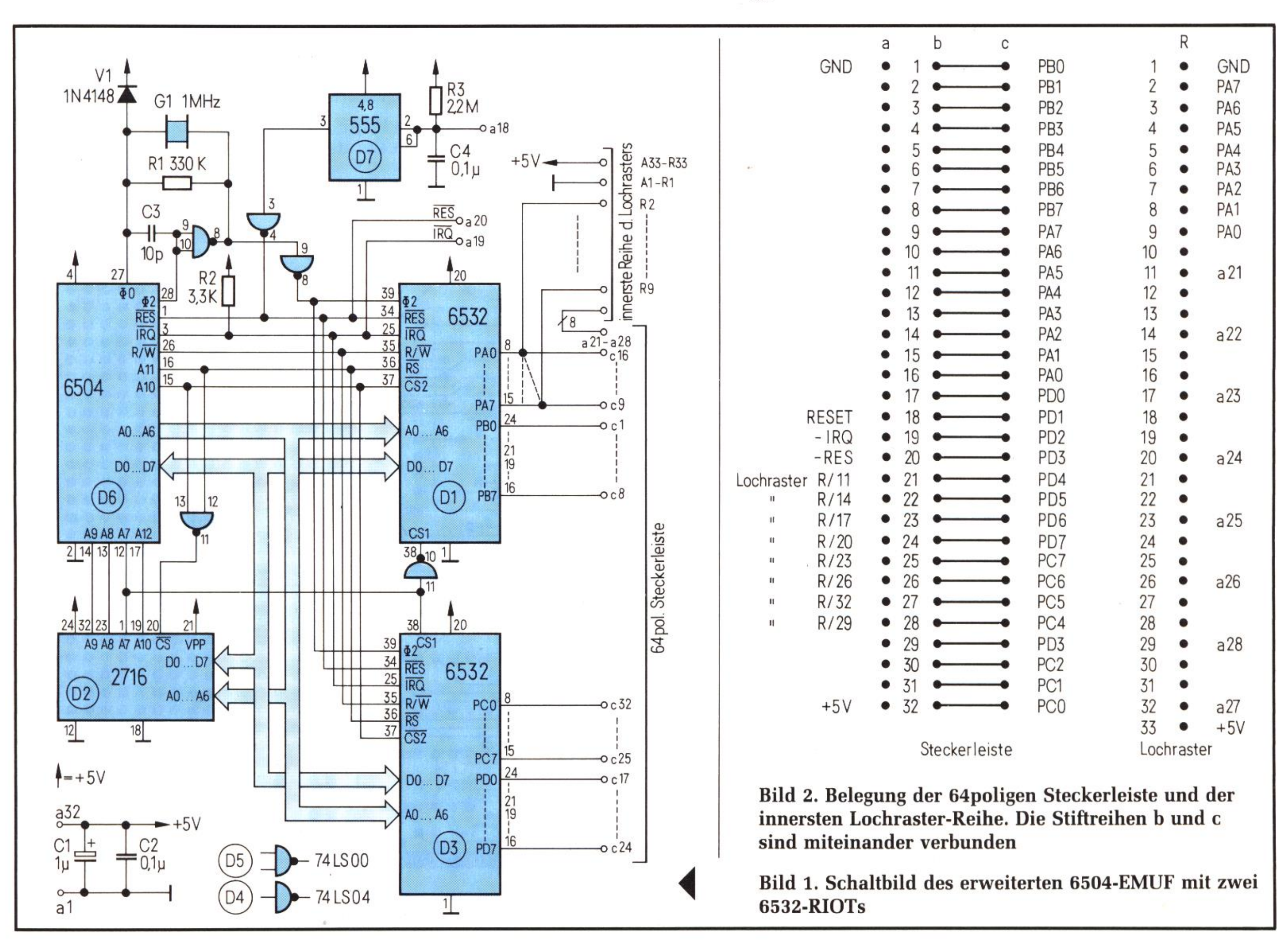


Bild 2. Belegung der 64poligen Steckerleiste und der innersten Lochraster-Reihe. Die Stiftreihen b und c sind miteinander verbunden

Bild 1. Schaltbild des erweiterten 6504-EMUF mit zwei 6532-RIOTs

Bild 3. Bauteilseite der Platine (Maßstab 1 : 1)

Wegen der verhältnismäßig großen Pakkungsdichte war es nicht zu vermeiden, daß verschiedentlich Leiterbahnen zwischen IC-Füßchen hindurch verlaufen. Deswegen muß bei der Bestückung der Platine unbedingt auf sauberes Arbeiten mit dem Lötkolben geachtet werden.

Bild 3 zeigt die Bauteilseite der Platine im Maßstab 1 : 1, *Bild 4* die Lötseite. Die Lage der Bauteile geht aus *Bild 5* hervor. Für die Kondensatoren ist ein Kombiraster vorgesehen (5 mm oder 7,5 mm). *Tabelle 2* enthält eine Liste aller Bauelemente.

Weder der AIM-65 noch der KIM-1 oder

Bild 4. Lötseite der Platine

Tabelle 1: Adressenbereiche des neuen „6504-EMUF mal 2“

Adresse	Bedeutung
$1FFF $1C00	Zweite Hälfte des EPROMs (entspricht den EPROM-Adressen $0400...$07FF)
$1BFF $1000	Wie $0000...$0BFF
$0FFF $0C00	Erste Hälfte des EPROMs (entspricht den EPROM-Adressen $0000...$03FF)
$089F $0880	Wie $0800...$081F, jedoch zweiter 6532 (PC, PD und Timer 2)
$081F $081C	Wie $0814...$0817, jedoch mit Interrupt
$0817 $0816 $0815 $0814	Timer 1, Takt × 1024 × 64 × 8 × 1
$0803 $0802 $0801 $0800	PB-Richtungsregister PB-Datenregister PA-Richtungsregister PA-Datenregister
$00FF $0080	RAM-Bereich des zweiten 6532 (IC 3), identisch mit $0180...$01FF
$007F $0000	RAM-Bereich des ersten 6532 (IC 1), identisch mit $0100...$017F

andere 6502-Computer, die als Entwicklungssystem in Frage kommen, verfügen über 32 I/O-Leitungen. Für die Entwicklung von Programmen, die mehr als zwei Ports benutzen, ist es also entweder notwendig, die Hardware des Entwicklungssystems entsprechend zu erweitern, oder beim Testen der EMUF-Programme ein gewisses Risiko einzugehen, das sich jedoch durch die folgenden Kniffe recht gering halten läßt.

○ Als dritter und vierter Port wird je eine RAM-Adresse benutzt, die durch eine entsprechende Interrupt-Service-Routine über Tastatur und Anzeige zugänglich gemacht wird.

○ Manchmal genügt es, bei Ausgängen einen „blinden Port“, d. h. eine RAM- oder ROM-Adresse zu benutzen. Dies gilt besonders für Unterprogramme, die bereits anderweitig ausgetestet worden sind.

Welch großer Anwendungsbereich einem EMUF zukommt, ist wohl am besten mit dem mc-EMUF-Sonderheft bewiesen worden. Ein EMUF mit doppelten Möglichkeiten erweitert das Spektrum sicher noch. Als Beispiele seien nur genannt: EPROM-Programmiergerät, universelles Interface für IEC-Bus, 16-Kanal-A/D-Wandler für den IEC-Bus usw. Alle diese Anwendungen sind wegen des Bedarfs an mehr als 16 I/O-Leitungen nicht mit dem alten 6504-EMUF

Tabelle 2: Bauteileliste

D1, D3	6532
D2	2716 (2532)
D4	74LS04
D5	74LS00
D6	6504
D7	NE 555
V1	1 N 4148
G1	Quarz 1 MHz
R1	330 kΩ
R2	3,3 kΩ
R3	2,2 MΩ
C1	1 µF/16 V (Elko)
C2, C4	0,1 µF (Keramik)
C3	10 pF (Keramik)

zu realisieren gewesen. Der Ausweg über den in etwa gleichwertigen Z80-EMUF (mc 4/1983) scheitert meistens an der „Fremdsprache“ und daran, daß eben auch ein Z80-Entwicklungssystem vorhanden sein muß.

Die Firmen Frank-Elektronik, Gugelstraße 129, 8500 Nürnberg, und Steinmetz-Elektronik, Nürnberger Straße 49, 8600 Bamberg, bieten einen Bausatz für unter 100 DM an. Er enthält alle Bauteile, die Platine und Sockel, jedoch nicht das EPROM und die 64polige Steckerleiste.

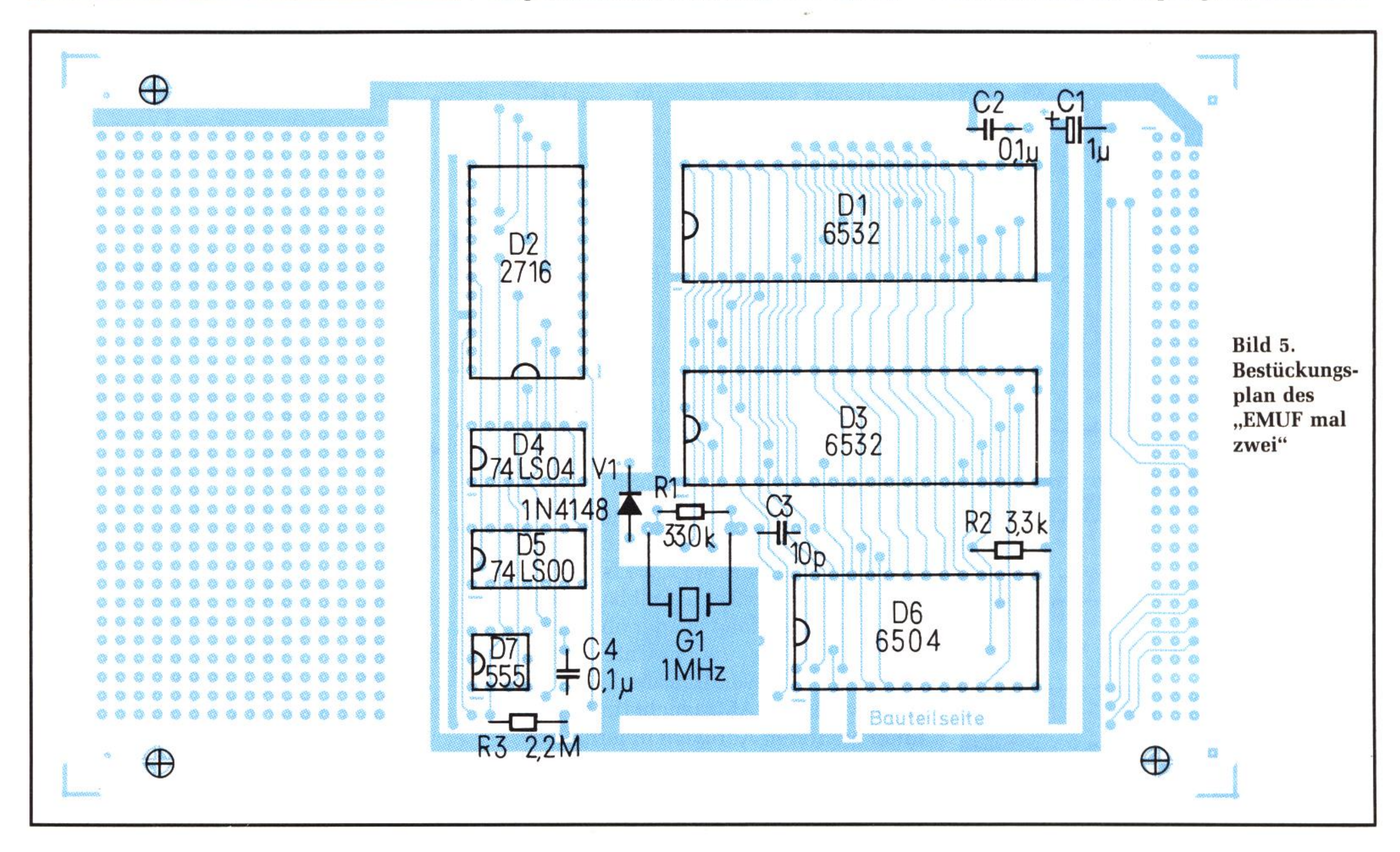


Bild 5. Bestückungsplan des „EMUF mal zwei“

Stephan Thienel, Thomas Sauer

Mehr Speicher – mehr Anwendungen

Ein 6502-EMUF mit viel Speicherplatz

Im Gegensatz zu den bisherigen 6504-EMUFs [1, 2] besitzt die hier vorgestellte Variante mehr Speicherplatz sowohl im RAM- als auch im EPROM-Bereich. Es sind maximal 8 KByte RAM sowie 16 KByte EPROM möglich! Statt dem RIOT-Baustein 6532 werden zwei VIAs 6522 eingesetzt. Anwenderspezifische Schaltungen finden auf einem Lochrasterfeld Platz.

Die Idee zum 6502-EMUF entstand aus dem Bedürfnis nach mehr Speicherplatz. Es sollte gleich sehr viel Speicherplatz mehr sein, deshalb wird anstelle des 6504 der große Bruder 6502 eingesetzt, was sich bei den Kosten eher günstig auswirkt. Außerdem kann jetzt der 6532 mit seinen 128 Byte RAM durch den komfortableren und billigeren VIA-Baustein 6522 ohne internes RAM ersetzt werden. „EMUF" heißt übrigens „Einplatinen-Mikrocomputer für universelle Festprogrammierung".

Adresse	Bereich	Bestückung
		2716 (2 kByte)
$F000		2732 (4 kByte)
$E000		2764 (8 kByte)
$C000	EPROM	27128 (16 kByte)
$A000	frei	
$8000	frei	
$6010	VIA 2 (IC9)	
$6000	VIA 1 (IC8)	
$4000	frei	
$2000	frei	
	RAM	6264 (8 kByte)
$0000		6116 (2 kByte)

Bild 2. So sieht die Adressenbelegung bei unterschiedlicher Bestückung aus

Varianten im Ausbau des Speichers

Die Adressierlogik wurde halb so aufwendig wie befürchtet. *Bild 1* zeigt den Schaltplan. Ein 74LS138 erzeugt aus den obersten drei Adreßleitungen die Selectsignale für acht Blöcke von 8 KByte. Sie heißen SEL-0/1, SEL-2/3,..., SEL-E/F. Die Aufteilung der Adressen ergibt sich aus *Bild 2*. Wegen der Zeropage muß das RAM im Block 0/1 liegen. Der dafür vorgesehene 28polige Sockel kann mit Hilfe einer Lötbrücke von 2 KByte (CMOS-RAM 6116) auf 8 KByte (CMOS-RAM 6264) umgeschaltet werden.
Ähnlich ist es beim EPROM-Sockel. Das Chip-Select-Signal wird mit zwei Gattern aus SEL-C/D und SEL-E/F abgeleitet. So ist es möglich, das 16-KByte-EPROM 27128 zu verwenden. Aber auch ein 2716, ein 2732 oder ein 2764 ist in dem vorgesehenen Sockel betreibbar. Die Umschaltung erfolgt wieder über eine einzige Lötbrücke auf der Unterseite der Platine. Bei Verwendung der kleineren EPROMs bzw. RAMs taucht ihr Inhalt unter mehreren Adressen innerhalb des entsprechenden Blocks auf.
Die unbenutzten Block-Select-Signale SEL-2/3, SEL-4/5, SEL-8/9 und SEL-A/B sind zusammen mit den Leitungen $\overline{IRQ}$, $\overline{NMI}$, RESET.IN und $\overline{RES}$ auf die innerste Reihe des Lochrasters herausgeführt. Entlang der VIA-Bausteine liegen ebenso deren Port- und Steueranschlüsse auf dem Lochraster. Auf die bisher übliche 64polige Steckerleiste wurde nicht ganz verzichtet. Wer sie unbedingt braucht, wird feststellen, daß das Lochraster ihren Einbau zuläßt. Sogar die Befestigungslöcher sind bereits vorhanden. Die Anschlüsse müssen allerdings verdrahtet werden.

Platz für Anwenderschaltungen

Die übrige Schaltung entspricht den Vorschlägen der 6502-Hersteller. *Bild 3* zeigt die Bauteil- bzw. Lötseite der Platine. An den achteckigen Lötaugen ist zu erkennen, daß die Filme aus einem CAD-System mit Fotoplotter stammen. Naturgemäß ist es mit Hobbymitteln kaum möglich, eine doppelseitige, durchkontaktierte Platine selbst mit der nötigen Präzision herzustellen; dem Hobbyisten sei daher dringend zum Kauf der fertig geätzten und gebohrten Platine geraten.

Die Bestückung geht aus *Bild 4* hervor, die erforderlichen Bauelemente sind in der *Tabelle* aufgeführt. Eine solche Packungsdichte auf der Platine hat leider zur Folge, daß die Leiterbahnen sehr eng geführt werden müssen. An einer Stelle verlaufen gleich zwei Bahnen gemeinsam zwischen zwei IC-Füßchen hindurch. Auch ließ es sich nicht vermeiden, daß einige wenige Verbindungen

Tabelle: Die Stückliste zum 6502-EMUF

Bauteil	Typ
IC1	74 LS 138
IC2	74 LS 00
IC3	74 LS 04
IC4	NE 555
IC5	6502 (65C02)
IC6	2716/32/64/128
IC7	6116/6264
IC8, 9	6522 (65C22)
R1	2,2 MΩ
R2	330 kΩ
R3, 4	3,3 kΩ
R5	560 Ω
C1	10 pF (ker.)
C2...5	0,1 μF
C6	10 μF (Tantal)
D1	1 N 4148
Q1	Quarz 1 MHz

das Lochraster überqueren. Es ist also unbedingt erforderlich, beim Löten äußerste Sauberkeit und Vorsicht walten zu lassen – eine gute Gelegenheit, die Lötspitze einmal wieder in Schuß zu bringen.
Die Frage nach einem passenden Entwicklungssystem kann nicht so einfach beantwortet werden. Ob C-64, CBM-8032, AIM-65 oder andere Mikrocomputer mit dem 6522, alle haben zu wenig I/O-Leitungen. In jedem Falle sind also entweder Erweiterungsarbeiten oder Tricks erforderlich. Trotzdem dürfte es einem einigermaßen erfahrenen 6502-Programmierer nicht schwer fallen, in kürzester Zeit ein kleines Testprogramm zum Laufen zu bringen.
Die Anwendungsmöglichkeiten des 6502-EMUF übersteigen weit die der Vorgänger. Hier einige denkbare Beispiele: Ein Druckerinterface mit 8 KByte Puffer entlastet den Rechner, in 16 KByte EPROM lassen sich zusätzlich mehre-

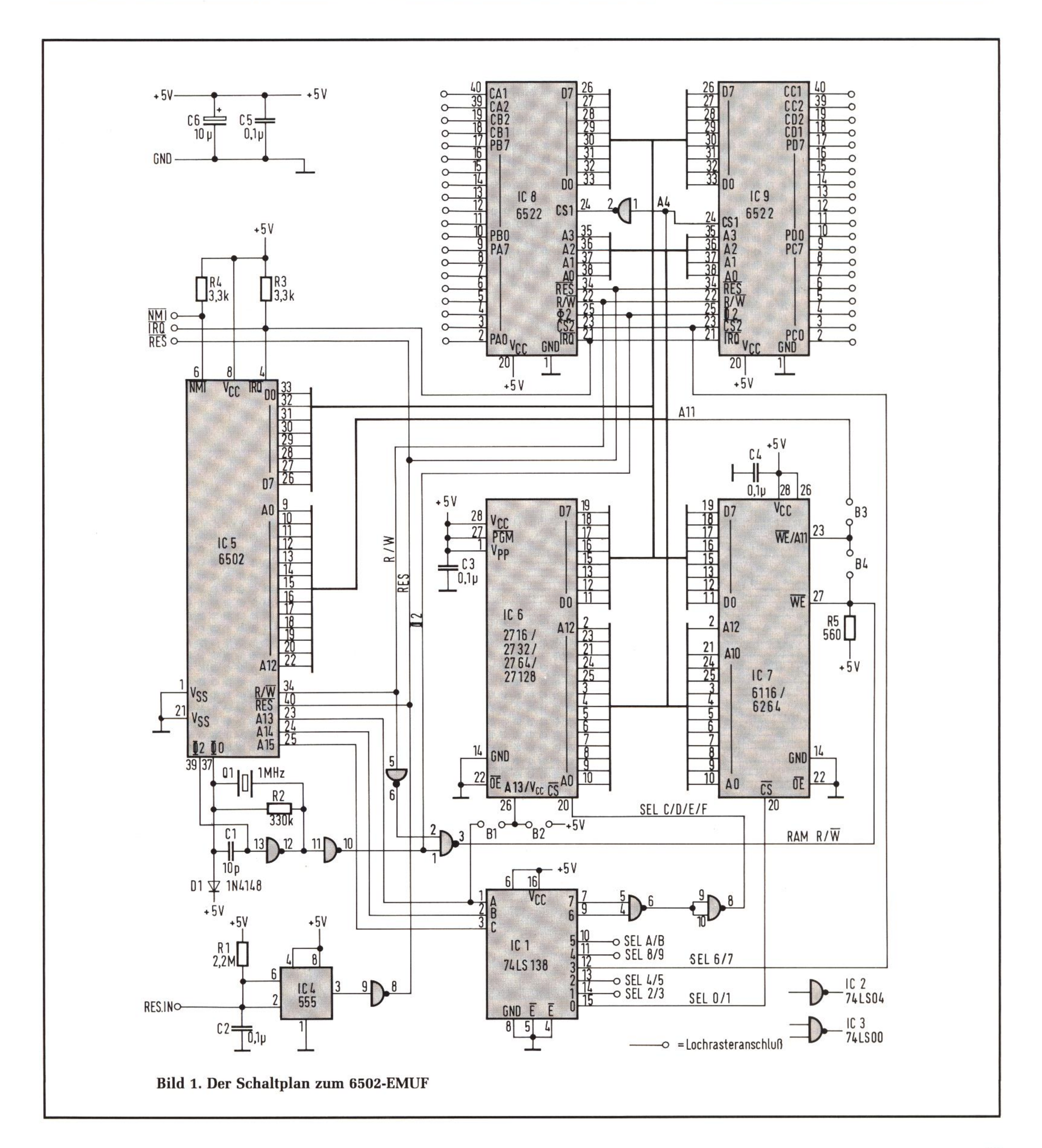


Bild 1. Der Schaltplan zum 6502-EMUF

STH13-C2

L39 BAUTEILSEITE

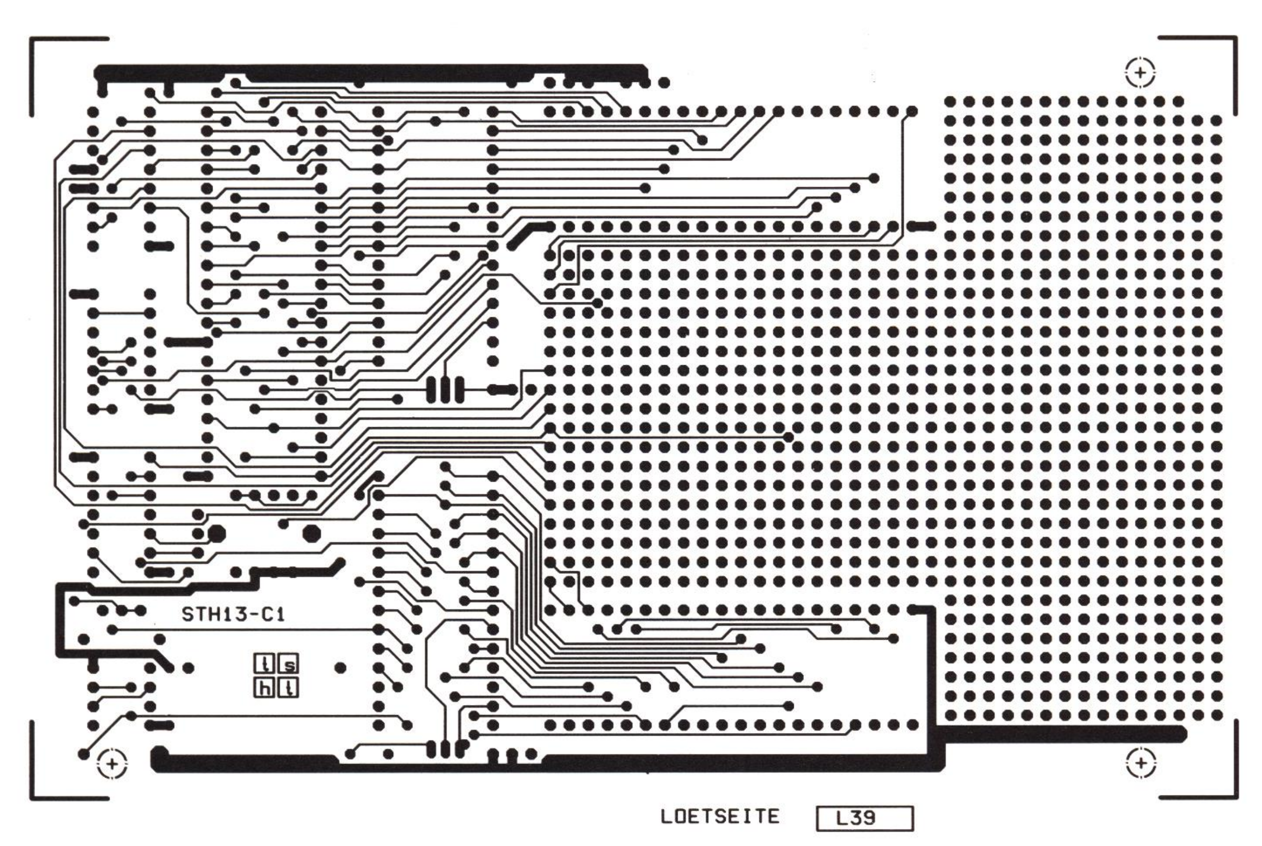


Bild 3. Das Layout der Platine mit der Bestückungsseite oben und der Lötseite unten

re Schriftarten unterbringen, z. B. auch solche, die mehrzeilige Größe besitzen. Zusammen mit einer Tastatur und einer sechsstelligen Anzeige könnte ein kleines, aber leistungsfähiges Entwicklungs- und Lehrsystem entstehen.
Die Firmen Frank-Elektronik, Matthiasstraße 3, 8500 Nürnberg, und Steinmetz-Elektronik, Nürnberger Straße 49, 8600 Bamberg, vertreiben einen Bausatz für unter 100 DM. Er enthält die Platine, alle Bauteile einschließlich der Sockel für die ICs sowie das CMOS-RAM 6116, aber keine Steckerleiste und kein EPROM. Zum ersten Mal gibt es auch eine CMOS-Version des Bausatzes mit 65C02 und 65C22. Sie eignet sich insbesondere für batteriebetriebene Geräte.

Literatur

[1] *Feichtinger, H.*: Mädchen für alles. mc 1981, Heft 2, Seite 20, und EMUF-Sonderheft, Seite 10.

[2] *Thienel, S.*: EMUF mal zwei. mc 1984, Seite 41.

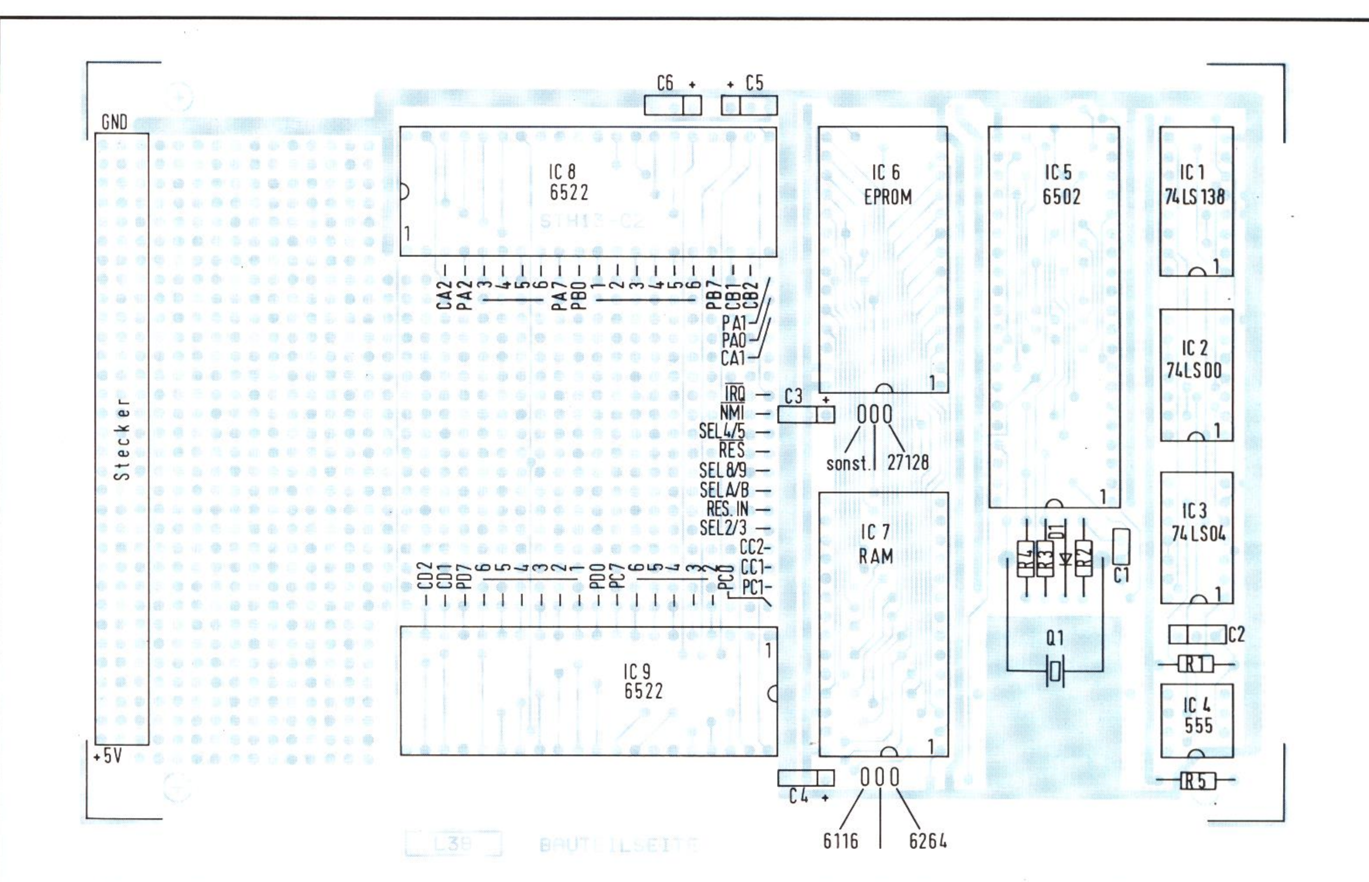


Bild 4. Der Bestückungsplan zeigt auch die Belegung der Lochrasteranschlüsse. Die Brücken zur Einstellung des RAM- bzw. EPROM-Typs befinden sich auf der Lötseite

Stephan Thienel, Thomas Sauer, Wolfgang Kostal

Entwicklungs-hilfe

Ein Monitorprogramm für den 6502-EMUF

Der 6502-EMUF [1] erhält zum Entwickeln und Testen von Programmen einen Terminal-Anschluß, ein Monitorprogramm mit 65C02-Disassembler, eine Speichererweiterung bis 24 KByte RAM und eine Single-Step-Einrichtung.

Kaum ein gängiger 6502-Mikrocomputer verfügt über so viele freie Ports wie der 6502-EMUF. Deswegen lassen sich Programme für ihn oft nur mit einem erheblichen Verschleiß an EPROMs und auch an Löschlampen-Lebensdauer entwikkeln. Bei 8 KByte RAM, die sogar, wie anschließend beschrieben, mühelos bis 24 KByte erweitert werden können, liegt es auf der Hand, den EMUF selbst als Entwicklungssystem zu verwenden. Alles was fehlt, ist ein Terminal und ein Monitorprogramm.

Das Terminal: Beispiel C-64

Das Monitorprogramm (*Bild 1*) enthält die Routinen für eine V.24-Schnittstelle, wie sie auch für den Anschluß eines Akustikkopplers verwendet wird. Als Terminal eignet sich also im Prinzip jeder Rechner, der auch einen Anschluß für ein Modem hat. *Bild 2* zeigt zum Beispiel, wie der EMUF mit der RS-232-Schnittstelle des C-64 von Commodore verbunden wird. Hier sind nicht einmal Pegelwandler notwendig, weil der User-Port des C-64 ebenfalls TTL-Signale liefert bzw. verarbeitet. Als Eingang am EMUF dient PC7, Ausgang ist PD0. In *Bild 3* ist ein kleines Kommunikationsprogramm für den C-64 aufgelistet. Es erklärt sich durch seine Kommentare selbst.

Das Monitorprogramm

Die Ein-/Ausgabe-Einheit ist ein seriell angeschlossener Terminal-Rechner. Das Problem der Übertragungsgeschwindigkeit wurde ähnlich wie einst beim legendären KIM1 von Commodore gelöst: das Terminal schickt als erstes Zeichen $FF, damit der EMUF durch Messung der Länge eines Startbits die Baudrate erkennen kann. Vor dem Übertragen des Bytes $FF sollte am EMUF unbedingt ein Reset ausgelöst werden. Es sind Übertragungsgeschwindigkeiten bis 2400 Baud und darüber hinaus möglich.

Die *Tabelle 1* zeigt eine Zusammenstellung der möglichen Befehle, die durchaus noch individuell erweitert werden können. Denkbar sind zum Beispiel Befehle zum Verschieben eines Speicherbereichs in einen anderen, Vergleichen und Füllen von Speicherbereichen usw. Die vorgestellte Befehlsliste wurde für den Betrieb an einem Terminalrechner ausgelegt, der über einen Assembler verfügt, so daß solche Routinen eigentlich nicht gebraucht werden und nur unnötig Speicherplatz belegen. Wichtiger sind in diesem Falle Routinen zum Übertragen von Hex-Code zwischen Terminalrechner und EMUF, damit ein assembliertes Programm zum Testen ins RAM des EMUF geschrieben werden kann. Dabei wird jedes Byte gemäß seiner Darstellung im Hex-Code in zwei ASCII-Zeichen zerlegt und dann erst übertragen. Der Befehl „L" dient zum Laden eines Programms vom Terminalrechner in den

```
0000   a2 ff 9a 86 f4 20 34 fa a9 ff 8d ef 00 2c 1f 60   08d2
0010   30 fb a9 fc 18 69 01 90 03 ee ef 00 ac 1f 60 10   06fd
0020   f3 8d ee 00 a2 08 20 b8 fa 20 26 fa 20 21 fa 20   0785
0030   ac f9 20 d6 f9 a0 00 b1 fb 20 b7 f9 20 d6 f9 a9   0a48
0040   00 85 f9 85 fa 20 aa fa 20 d8 f9 48 20 8a fa 68   0906
0050   a2 11 dd 6c f8 f0 06 ca 10 f8 4c 45 f8 8a 0a aa   0883
0060   bd 7e f8 a8 e8 bd 7e f8 48 98 48 60 4e 2b 2d 2e   0852
0070   52 47 4d 4c 53 2a 7f 00 6e 67 6d 6c 73 72 a1 f8   065a
0080   a7 f8 ad f8 bb f8 c4 f8 05 f9 18 f9 6c f9 8d f9   0bad
0090   d9 fd 2b f8 2b f8 a1 f8 05 f9 18 f9 6c f9 8d f9   0aaf
00a0   c4 f8 20 e2 fa 4c 2c f8 20 db fa 4c 2c f8 38 a5   096a
00b0   fb e9 01 85 fb b0 02 c6 fc 4c 2c f8 a0 00 a5 f9   0987
00c0   91 fb 4c a8 f8 20 13 fa a2 03 20 d6 f9 b5 f4 20   0902
00d0   b7 f9 ca 10 f5 20 d6 f9 a5 f2 20 b7 f9 a5 f1 20   0a8b
00e0   b7 f9 20 d6 f9 20 d6 f9 20 d6 f9 20 d6 f9 a2 08   0a16
00f0   a5 f3 85 e3 06 e3 a9 30 69 00 20 d8 f9 ca d0 f4   09aa
0100   20 21 fa 4c a8 f8 a6 f4 9a a5 fc 48 a5 fb 48 a5   09d1
0110   f3 48 a6 f6 a4 f5 a5 f7 40 a9 02 85 e3 20 21 fa   099a
0120   20 eb fa 20 ac f9 20 d6 f9 a2 08 86 e9 a0 00 b1   0923
0130   fb 20 b7 f9 20 d6 f9 20 db fa c6 e9 d0 ef 20 f4   0b31
0140   fa 20 d6 f9 a2 08 86 e9 a0 00 b1 fb c9 20 30 0a   0871
0150   c9 7b 10 06 20 d8 f9 4c 5f f9 a9 2e 20 d8 f9 20   07d7
0160   db fa c6 e9 d0 e2 c6 e3 d0 b3 4c 2c f8 20 a3 f9   0b8e
0170   b0 19 a5 f9 48 a5 fa 48 20 7b fa a0 00 91 fb 68   08bf
0180   85 fa 68 85 f9 20 db fa 4c 6d f9 4c 2c f8 20 a3   093f
0190   f9 b0 0d a0 00 b1 fb 20 b7 f9 20 db fa 4c 8e f9   099a
01a0   4c 2c f8 a5 fb c5 f9 a5 fc e5 fa 60 a5 fc 20 b7   0b26
01b0   f9 a5 fb 20 b7 f9 60 85 f8 4a 4a 4a 4a 20 c8 f9   094f
01c0   a5 f8 20 c8 f9 a5 f8 60 29 0f c9 0a 18 30 02 69   0739
01d0   07 69 30 4c d8 f9 a9 20 85 ff 48 86 fd 84 fe 20   0877
01e0   51 fa ad 10 60 29 fe 8d 10 60 20 51 fa a2 08 ad   074e
01f0   10 60 29 fe 4e ff 00 69 00 8d 10 60 20 51 fa ca   067f
0200   d0 ed ad 10 60 09 01 8d 10 60 20 51 fa a6 fd a4   0793
0210   fe 68 60 a2 00 bd 46 ff 20 d8 f9 e8 e0 20 d0 f5   0a08
0220   60 a9 0d 4c d8 f9 a2 00 bd 30 ff 20 d8 f9 e8 e0   097a
0230   16 d0 f5 60 d8 78 a2 00 8e 13 60 a2 01 8e 12 60   06d1
0240   8e 10 60 a9 0e 85 ea 85 ec a9 ff 85 eb 85 ed 58   0977
0250   60 ad ef 00 8d f0 00 ad ee 00 38 e9 01 b0 03 ce   07b7
0260   f0 00 ac f0 00 10 f3 60 ad ef 00 8d f0 00 ad ee   08a3
0270   00 4a 4e f0 00 90 e3 09 80 b0 e0 20 aa fa 20 8a   0782
0280   fa 20 aa fa 20 8a fa a5 f9 60 c9 30 30 1b c9 47   08b4
0290   10 17 c9 40 30 03 18 69 09 2a 2a 2a 2a a0 04 2a   0363
02a0   26 f9 26 fa 88 d0 f8 a9 00 60 86 fd 84 fe a2 08   0947
02b0   2c 1f 60 30 fb 20 51 fa 20 68 fa ad 1f 60 29 80   0698
```

Bild 1. Das Monitorprogramm fürs EPROM, die Adresse $0000 im EPROM entspricht $F800 im EMUF. Das Programm ist also im EPROM, falls dieses größer als 2 KByte ist, immer in den obersten 2 KByte unterzubringen



```
02c0  4e ff 00 0d ff 00 8d ff 00 20 51 fa ca d0 ec 20  07f6
02d0  68 fa a5 ff a6 fd a4 fe 4a 2a 60 e6 fb d0 02 e6  0ab8
02e0  fc 60 a5 f9 85 fb a5 fa 85 fc 60 a5 fb 85 e7 a5  0bab
02f0  fc 85 e8 60 a5 e7 85 fb a5 e8 85 fc 60 00 08 0a  0955
0300  18 1a 28 2a 38 3a 40 48 4a 58 5a 60 68 6a 78 7a  049e
0310  88 8a 98 9a a8 aa b8 ba c8 ca d8 da e8 ea f8 fa  0c10
0320  10 30 50 70 80 90 b0 d0 f0 09 29 49 69 89 a0 a2  072f
0330  a9 c0 c9 e0 e9 04 05 06 14 24 25 26 45 46 64 65  05e1
0340  66 84 85 86 a4 a5 a6 c4 c5 c6 e4 e5 e6 01 21 41  0945
0350  61 81 a1 c1 e1 11 31 51 71 91 b1 d1 f1 15 16 34  078c
0360  35 36 55 56 74 75 76 94 95 b4 b5 d5 d6 f5 f6 96  0933
0370  b6 12 32 52 72 92 b2 d2 f2 0c 0d 0e 1c 20 2c 2d  0582
0380  2e 4c 4d 4e 6d 6e 8c 8d 8e 9c ac ad ae cc cd ce  08a1
0390  ec ed ee 1d 1e 3c 3d 3e 5d 5e 7d 7e 9d 9e bc bd  0823
03a0  dd de fd fe 19 39 59 79 99 b9 be d9 f9 6c 7c 3f  09e3
03b0  3f 3f 42 52 4b 50 48 50 41 53 4c 43 4c 43 49 4e  048e
03c0  43 50 4c 50 52 4f 4c 53 45 43 44 45 43 52 54 49  04b2
03d0  50 48 41 4c 53 52 43 4c 49 50 48 59 52 54 53 50  04dc
03e0  4c 41 52 4f 52 53 45 49 50 4c 59 44 45 59 54 58  04e4
03f0  41 54 59 41 54 58 53 54 41 59 54 41 58 43 4c 56  04ee
0400  54 53 58 49 4e 59 44 45 58 43 4c 44 50 48 58 49  04dc
0410  4e 58 4e 4f 50 53 45 44 50 4c 58 42 50 ^c 42 4d  04d0
0420  49 42 56 43 42 56 53 42 52 41 42 43 43 42 43 53  0484
0430  42 4e 45 42 45 51 4f 52 41 41 4e 44 45 4f 52 41  0489
0440  44 43 42 49 54 4c 44 59 4c 44 58 4c 44 41 43 50  049b
0450  59 43 4d 50 43 50 58 53 42 43 54 53 42 4f 52 41  04c7
0460  41 53 4c 54 52 42 42 49 54 41 4e 44 52 4f 4c 45  04ac
0470  4f 52 4c 53 52 53 54 5a 41 44 43 52 4f 52 53 54  04f5
0480  59 53 54 41 53 54 58 4c 44 59 4c 44 41 4c 44 58  04e2
0490  43 50 59 43 4d 50 44 45 43 43 50 58 53 42 43 49  ^4a4
04a0  4e 43 4f 52 41 41 4e 44 45 4f 52 41 44 43 53 54  49b
04b0  41 4c 44 41 43 4d 50 53 42 43 4f 52 41 41 4e 44  047f
04c0  45 4f 52 41 44 43 53 54 41 4c 44 41 43 4d 50 53  049a
04d0  42 43 4f 52 41 41 53 4c 42 49 54 41 4e 44 52 4f  049a
04e0  4c 45 4f 52 4c 53 52 53 54 5a 41 44 43 52 4f 52  04df
04f0  53 54 59 53 54 41 4c 44 59 4c 44 41 43 4d 50 44  04c6
0500  45 43 53 42 43 49 4e 43 53 54 58 4c 44 58 4f 52  04c2
0510  41 41 4e 44 45 4f 52 41 44 43 53 54 41 4c 44 41  047b
0520  43 4d 50 53 42 43 54 53 42 4f 52 41 41 53 4c 54  04b7
0530  52 42 4a 53 52 42 49 54 41 4e 44 52 4f 4c 4a 4d  04b9
0540  50 45 4f 52 4c 53 52 41 44 43 52 4f 52 53 54 59  04e2
0550  53 54 41 53 54 58 53 54 5a 4c 44 59 4c 44 41 4c  04ee
0560  44 58 43 50 59 43 4d 50 44 45 43 43 50 58 53 42  04b4
0570  43 49 4e 43 4f 52 41 41 53 4c 42 49 54 41 4e 44  0491
0580  52 4f 4c 45 4f 52 4c 53 52 41 44 43 52 4f 52 53  04d2
0590  54 41 53 54 5a 4c 44 59 4c 44 41 43 4d 50 44 45  04b9
05a0  43 53 42 43 49 4e 43 4f 52 41 41 4e 44 45 4f 52  0490
05b0  41 44 43 53 54 41 4c 44 41 4c 44 58 43 4d 50 53  049c
05c0  42 43 4a 4d 50 4a 4d 50 23 7c b2 24 2d 39 51 59  04d8
05d0  61 73 75 7d 97 a8 b1 b2 b3 b4 20 d6 f9 a0 00 b1  090f
05e0  fb a2 b2 dd fc fa f0 03 ca d0 f8 86 e3 b9 c8 fd  0c8e
05f0  c8 c5 e3 90 f8 84 e4 a0 03 a9 af a2 fb 18 65 e3  0a58
0600  90 01 e8 88 d0 f7 85 e5 86 e6 b1 e5 20 d8 f9 c8  0aed
0610  c0 03 d0 f6 20 d6 f9 a2 00 a5 e3 dd cb fd 90 03  09da
0620  20 35 fe a2 00 18 a5 fb 65 e4 85 fb a5 fc 69 00  0880
0630  85 fc 4c 2c f8 e8 dd cb fd b0 fa 8a 0a aa bd f2  0b15
0640  fe a8 e8 bd f2 fe 48 98 48 60 a0 01 18 b1 fb a4  09cc
0650  fc aa 10 01 88 65 fb 90 01 c8 18 69 02 aa 90 01  06b6
0660  c8 4c 72 fe a9 24 4c d8 f9 a0 01 b1 fb aa c8 b1  09de
0670  fb a8 20 64 fe 98 20 b7 f9 8a 4c b7 f9 a9 23 20  08ff
0680  d8 f9 4c 8a fe a9 28 20 d8 f9 20 64 fe a0 01 b1  093b
0690  fb 4c b7 f9 20 85 fe 20 a2 fe 4c 9d fe a9 29 4c  095f
06a0  d8 f9 a9 2c 20 d8 f9 a9 58 4c d8 f9 a9 2c 20 d8  0982
06b0  f9 a9 59 4c d8 f9 20 85 fe 20 9d fe 4c ac fe 20  098c
06c0  8a fe 4c a2 fe 20 8a fe 4c ac fe 20 85 fe 4c 9d  099e
06d0  fe 20 69 fe 4c a2 fe 20 69 fe 4c ac fe 20 ec fe  09f8
06e0  4c 9d fe 20 ec fe 20 a2 fe 4c 9d fe a9 28 20 d8  0961
06f0  f9 4c 69 fe 49 fe 7c fe 89 fe 93 fe b5 fe be fe  0bf4
0700  c4 fe ca fe 68 fe d0 fe d6 fe dc fe e2 fe 85 f7  0dc8
0710  68 85 f3 68 85 f1 85 fb 68 85 f2 85 fc 84 f5 86  0a9d
0720  f6 ba 86 f4 20 34 fa 4c c5 f8 6c ea 00 6c ec 00  092f
0730  0d 0a 45 4d 55 46 20 49 49 49 20 2d 20 4d 4f 4e  0396
0740  49 54 4f 52 0d 0a 0d 0a 20 41 43 20 58 52 20 59  0353
0750  52 20 53 50 20 50 43 20 20 20 50 53 3a 4e 56 20  03c9
0760  42 44 49 5a 43 0d 0a 4c 6a c4 f0 07 4c 7d ac 68  05d1
0770  4c 7d ac a5 49 ee 11 c7 4c be ad 20 2e af ad 1e  07a8
0780  c7 d0 e9 a9 0f 4c 59 ae a2 0c 20 fb ae 4c 7d ac  0877
0790  a2 f6 20 ab a6 aa b9 35 c7 c9 2a d0 0a c8 20 a7  08c4
07a0  b1 f0 3a a2 3f d0 1a c9 28 d0 0f c8 20 a7 b1 a2  0858
07b0  fb 20 ab a6 a2 34 d0 09 d0 07 20 a7 b1 f0 0c a2  0808
07c0  4a 20 0a bd aa 4c 44 ab e8 e8 e8 e8 20 fb ae a5  0924
07d0  49 20 2e af a5 4a 4c 7b ae e8 e8 e8 e8 e8 e8 e8  0a02
07e0  e8 e8 20 fb ae a5 49 20 2e af a5 4a f0 0a a2 00  080f
07f0  ad 13 c7 f0 03 20 60 a4 4c 7e 2a ff 00 f8 2d ff  07b5
```

EMUF. Bei „S" dagegen (S ≙ save) wird ein Speicherbereich des EMUF im gleichen Verfahren an den Terminalrechner gesendet. Weil Hex-Code übertragen wird, kann man während der Übertragung direkt mitlesen.
Die wichtigsten Monitor-Einsprünge sind in *Tabelle 2* zusammengestellt. In der Zeropage werden lediglich die Zellen von $E3 bis $FF benutzt. Die Zellen $EA/EB dienen als NMI-Zeiger (für Single-Step auf $FF0E voreingestellt), $EC/ED enthalten den IRQ-Vektor (für Breaks auch auf $FF0E gesetzt).

Eine RAM-Speichererweiterung

Der 6502-EMUF kann bis zu 8 KByte RAM und 16 KByte EPROM ansprechen. Soll die Entwicklung auch der größtmöglichen Programme unterstützt werden, so sind zusätzlich 16 KByte RAM notwendig. Bei der Konzeption des EMUF wurde dies bereits berücksichtigt und die Ports auf die Adressen ab $6000 gelegt. So sind insgesamt 24 KByte für RAM frei. Es gibt auch keine Platz- oder Dekodierschwierigkeiten, da die drei RAM-Bausteine 6264 einfach übereinandergelötet werden können. Das erste RAM steckt dabei ganz normal im entsprechenden Sockel. Das nächste wird einfach mit allen Anschlüssen (bis auf Pin 20) auf das erste gelötet. Lediglich Pin 20 muß am Lochraster mit SEL 2/3 verbunden werden. Genauso verfährt man mit dem dritten RAM-IC. Sein Anschluß 20 kommt an SEL 4/5.

Single-Step-Betrieb

Der Anschluß SYNC am 6502 liefert nach dem Abarbeiten eines Maschinenbefehls ein Signal. Mit ihm kann ein NMI ausgelöst werden, der z. B. die Registerinhalte anzeigen kann. Der NMI-Zeiger bei $EA/EB wird beim Kaltstart bereits auf die entsprechende Routine bei $FF0E voreingestellt. Leider würde ein solcher Interrupt auch seine eigene Interrupt-Service-Routine anhalten. Des-

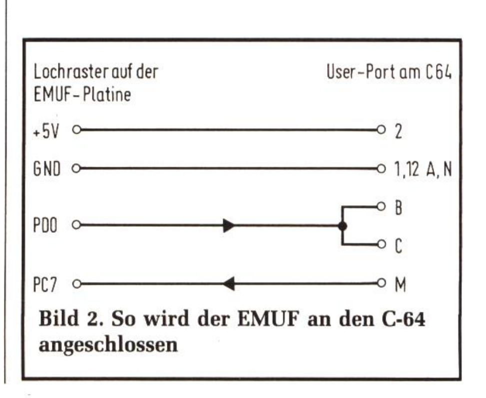


Bild 2. So wird der EMUF an den C-64 angeschlossen

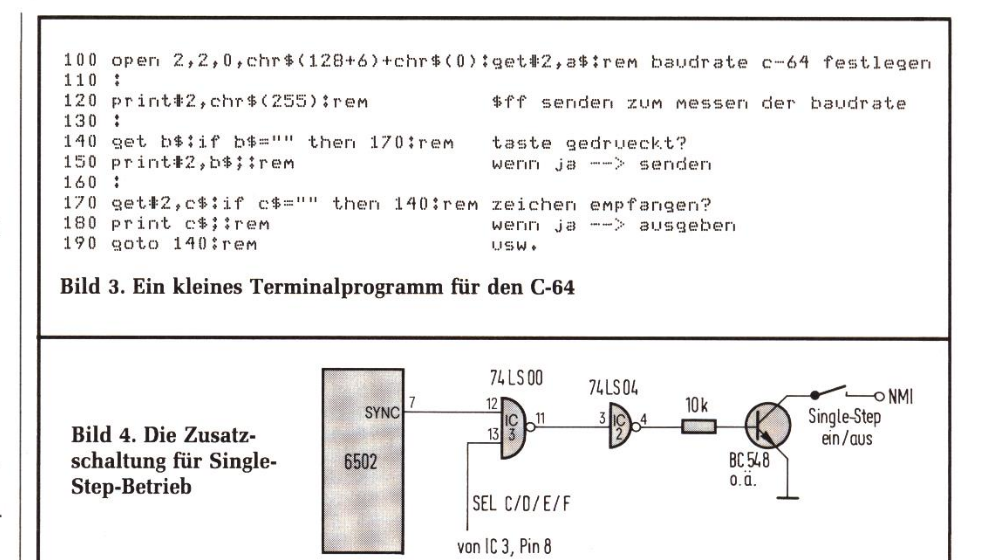
wegen wird in der kleinen Zusatzschaltung in *Bild 4* der Interrupt immer dann hardwaremäßig (mit SEL C/D/E/F) verhindert, wenn der Befehl im EPROM liegt (die Schaltung macht übrigens von den freien Gattern der EMUF-Schaltung Gebrauch). Es ist kein großer Nachteil, daß damit nur noch Programme im RAM im Single-Step-Verfahren abgearbeitet werden können.

Die Entwicklung eines EMUF-Programms

Nach der Entwicklung eines ausführlichen Flußdiagramms wird man mit dem Schreiben des Programms in Assembler beginnen. Dies geschieht auf dem Terminalrechner. Nach der Übersetzung liegt das assemblierte Maschinenprogramm dann irgendwo im RAM. Man wird es aus Sicherheitsgründen erst einmal abspeichern und dann ins RAM des EMUF übertragen. Dies kann zum Beispiel auch mit einem kleinen Basic-Programm geschehen. Jetzt kann der Test beginnen. Verläuft der erste Start nicht gleich erfolgreich und dafür sprechen alle Statistiken, so können spezielle Programmteile mit Breaks und Single-Step genau untersucht werden. Hier kann auch der eingebaute 65C02-Disassembler von großem Nutzen sein. Ist das Programm entwanzt und das Source-Programm entsprechend auf den neuesten Stand gebracht, dann ändert man die Startadresse des Programms so, daß es im EPROM-Bereich des EMUF liegt. Nach der Übersetzung muß es nur noch in ein EPROM gebrannt und dieses in den EMUF gesteckt werden.

Literatur

[1] *Thienel, S., Sauer, T.*: Mehr Speicher – mehr Anwendungen. mc 1985, Heft 5, Seite 96.



```
100 open 2,2,0,chr$(128+6)+chr$(0):get#2,a$:rem baudrate c-64 festlegen
110 :
120 print#2,chr$(255):rem           $ff senden zum messen der baudrate
130 :
140 get b$:if b$="" then 170:rem   taste gedrueckt?
150 print#2,b$;:rem                 wenn ja --> senden
160 :
170 get#2,c$:if c$="" then 140:rem zeichen empfangen?
180 print c$;:rem                   wenn ja --> ausgeben
190 goto 140:rem                    usw.
```

Bild 3. Ein kleines Terminalprogramm für den C-64

Bild 4. Die Zusatzschaltung für Single-Step-Betrieb

Tabelle 1. Die Liste der Monitorbefehle

XXXXN	aktuelle Adresse auf XXXX setzen
+	aktuelle Adresse um eins erhöhen
–	aktuelle Adresse um eins erniedrigen
XX.	Byte XX in aktuelle Adresse übernehmen
*	Befehl in aktueller Adresse disassemblieren
R	Registerinhalte anzeigen
M	Inhalte der nächsten 16 Speicherstellen anzeigen (Hex- und ASCII-Format)
G	Programm ab aktueller Adresse starten
XXXXL	Hex-Code von aktueller Adresse bis zur Zelle XXXX einlesen
XXXXS	Hex-Code von aktueller Adresse bis XXXX senden

Tabelle 2. Die Systemeinsprünge

$F9D8	OUTCH	Akku als 8 Bits ausgeben
$F9D6	OUTSP	Blank ausgeben
$FA21	CRLF	CR und LF ausgeben
$F9B7	PRTBYT	Akku als zwei ASCII-Zeichen ausgeben
$F9AC	PRTPNT	Inhalt von $00FB/FC als Adresse (vier ASCII-Zeichen) ausgeben
$FAAA	GETCH	8 Bits in Akku einlesen
$FA7B	GETBYT	zwei ASCII-Zeichen als Byte in den Akku einlesen
$F800	COLD	Kaltstart
$F82C	WARM	Warmstart
$FF0E	NMIV	NMI-Vektor für Register-Retten (wird auch in $00EC/ED als Break-Vektor benutzt)

Stephan Thienel, Thomas Sauer

Der EMUF-232

Die neueste Entwicklung der Bauteilpreise macht's möglich: Der in [1] vorgestellte 6502-EMUF erhält eine RS232-Schnittstelle, 32-KByte-RAM und – endlich – zwei 64polige VG-Steckerleisten mit herausgeführten Bus- bzw. Port- und Schnittstellenleitungen. Eine Lochrasterfläche steht nach wie vor zur Verfügung. Auch das Monitorprogramm aus Heft 10/86 [2] wurde angepaßt.

Die Schaltung wurde um einen ACIA 6551 ergänzt, mit dem eine komfortable RS232- bzw. V.24-Schnittstelle realisiert werden kann. Ein MAX232 übernimmt die nötige Pegelwandlung und die Erzeugung der ±10 V aus der EMUF-Betriebsspannung von 5 V. Leider war es nötig, eine Quarzfrequenz von 1,8432 MHz zu wählen. Damit sind dann die üblichen Baudraten per Software einstellbar. Die Erzeugung bestimmter Zeitintervalle durch die Timer in den VIAs dürfte dadurch nur unwesentlich erschwert sein. Programme, die für den alten EMUF konzipiert sind, müssen hier angepaßt werden! Anstelle des 2- bzw. 8-KByte-RAMs wird jetzt ein 32-KByte-RAM verwendet. Dabei gewinnt man nicht nur Speicherplatz, sondern auch die Möglichkeit, die Select-Leitungen des 74LS138 auf den Bereich von $8000 bis $BFFF zu verteilen. Das Chip-select-Signal des EPROMs (jetzt nur noch ein 27128) kommt über Gatter zustande (*Bild 1*).

Adresse	Belegung
$FFFF – $C000	16-KByte-EPROM 27128
$B800	frei
$B000	frei
$A800	frei
$A000	frei
$9800	frei
$9000	frei
$8800	ACIA
$8000	VIA1(0–F) VIA2(10–1F)
$7FFF – $0000	32-KByte-RAM 55256

Bild 2. So ist der Speicher des Systems eingeteilt

64 pol. Bus-Steckerleiste X1

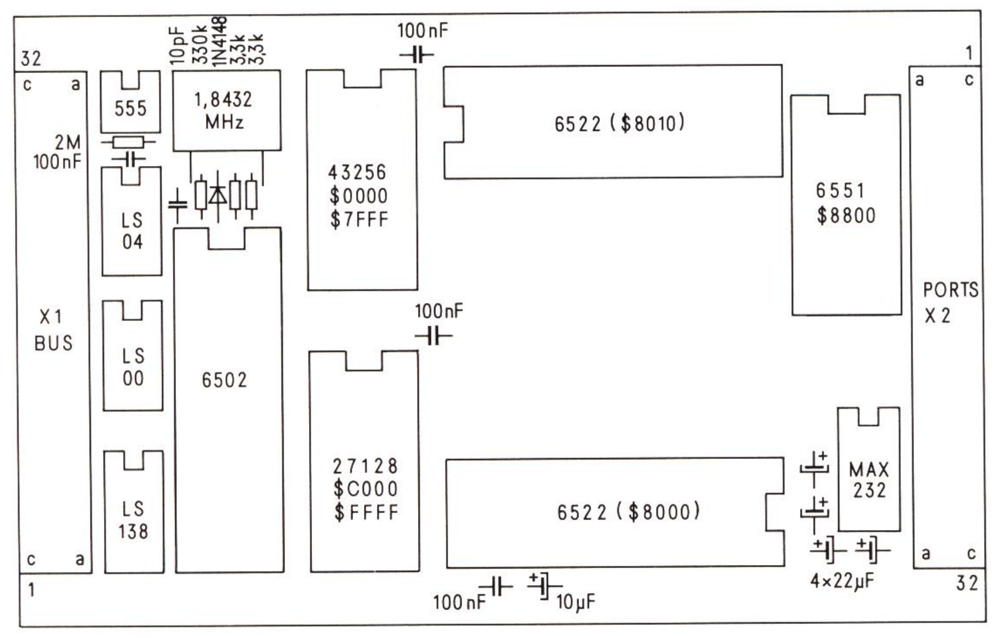
c		a
GND	+ 32 +	GND
nc	+ 31 +	RAM R/W̄
nc	+ 30 +	Φ1
R/W̄	+ 29 +	nc
SYNC	+ 22 +	nc
nc	+ 27 +	Φ2
A0	+ 26 +	A1
A2	+ 25 +	A3
A4	+ 24 +	A5
A6	+ 23 +	A7
A8	+ 22 +	A9
A10	+ 21 +	A11
A12	+ 20 +	A13
A14	+ 19 +	A15
nc	+ 18 +	nc
+10 V	+ 17 +	nc
GND	+ 16 +	GND
nc	+ 15 +	nc
nc	+ 14 +	nc
SEL $98	+ 13 +	nc
NMI	+ 12 +	IRQ
SEL $A8	+ 11 +	nc
D6	+ 10 +	D7
D4	+ 9 +	D5
D2	+ 8 +	D3
D0	+ 7 +	D1
SEL $90	+ 6 +	nc
RDY	+ 5 +	RES
GND	+ 4 +	GND
−10 V	+ 3 +	SEL $A0
SEL $B0	+ 2 +	SEL $B8
+5 V	+ 1 +	+5 V

64pol. Port-Steckerleiste X2

a		c
+5 V	+ 1 +	+5 V
nc	+ 2 +	nc
nc	+ 3 +	−10 V
GND	+ 4 +	GND
CD1	+ 5 +	CD2
PD6	+ 6 +	PD7
PD4	+ 7 +	PD5
PD2	+ 8 +	PD3
PD0	+ 9 +	PD1
PC6	+ 10 +	PC7
PC4	+ 11 +	PC5
PC2	+ 12 +	PC3
PC0	+ 13 +	PC1
CC2	+ 14 +	CC1
nc	+ 15 +	nc
GND	+ 16 +	nc
−DTR (TTL)	+ 17 +	+10 V
CB2	+ 18 +	CB1
PB7	+ 19 +	PB6
PB5	+ 20 +	PB4
PB3	+ 21 +	PB2
PB1	+ 22 +	PB0
PA7	+ 23 +	PA6
PA5	+ 24 +	PA4
PA3	+ 25 +	PA2
PA1	+ 26 +	PA0
CA1	+ 27 +	CA2
nc	+ 28 +	nc
nc	+ 29 +	nc
R×D	+ 30 +	T×D
CTS	+ 31 +	RTS
GND	+ 32 +	GND

Bild 3. Anschlußbelegung der beiden VG-Leisten ▸

Bild 4. Bestückung der Platine ▾



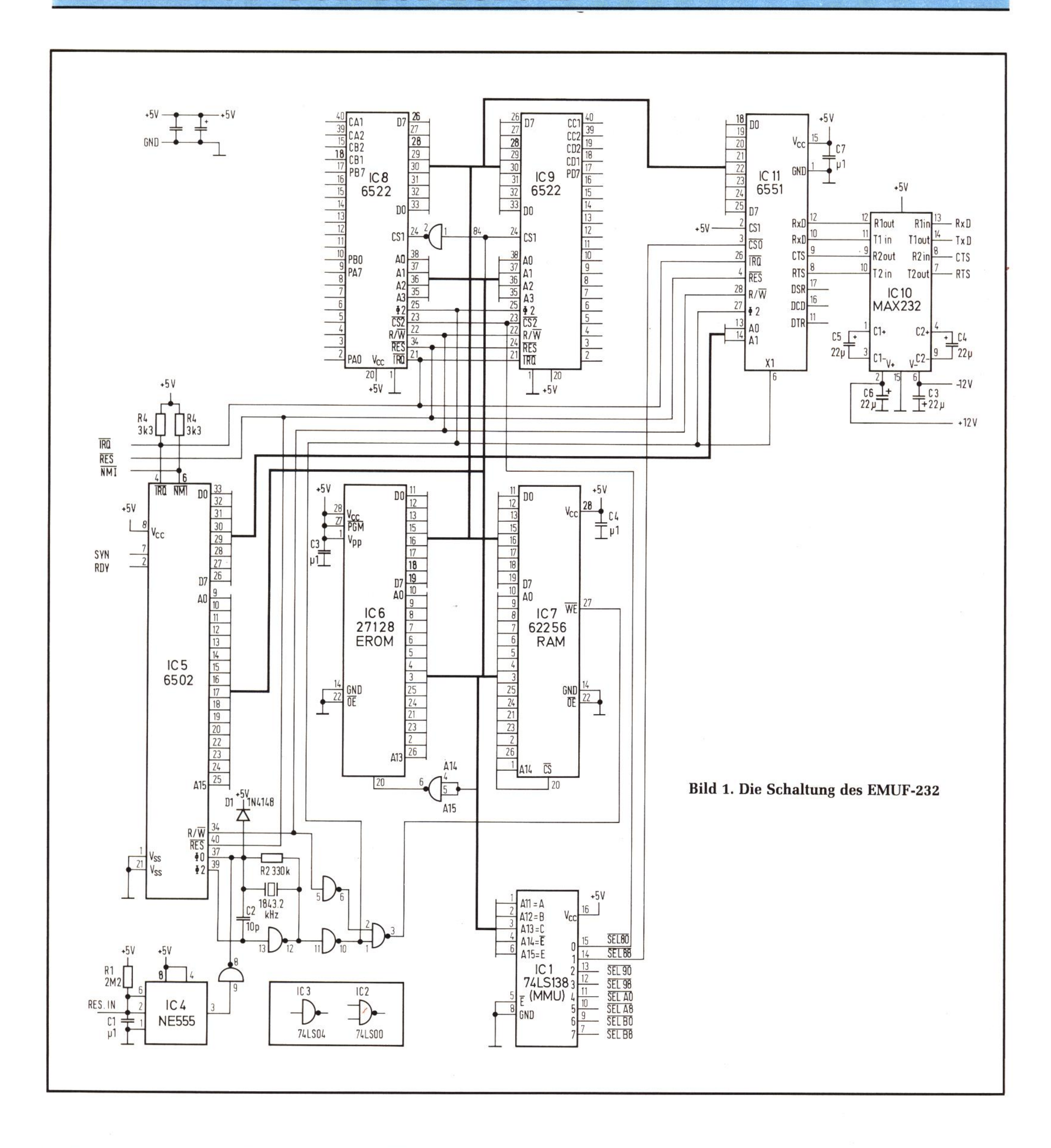


Bild 1. Die Schaltung des EMUF-232

Die Speicherverteilung ergibt sich aus *Bild 2*. Leider liegen die VIAs nicht mehr ab Adresse $6000, sondern bei $8000 und $8010, so daß ein eventuell vom alten EMUF her vorhandenes Programm auch in diesem Punkt angepaßt werden muß.

Die Platine erhielt an ihren Schmalseiten je einen Anschluß für eine 64polige VG-Leiste (*Bild 3*). Auf der einen Seite sind sämtliche Bus-Leitungen und die freien Select-Leitungen herausgeführt. Die andere Steckleiste macht alle 32 Port- und Handshake-Leitungen der zwei VIAs und die Anschlüsse der V.24-Schnittstelle zugänglich.

Bezugsquellen für Platine und Bausatz sind die Firmen Frank Elektronik, Nürnberg, Steinmetz Elektronik, Bamberg, sowie Elektronikladen, Detmold.

Literatur

[1] *Thienel, Stephan; Sauer, Thomas:* Mehr Speicher – mehr Anwendungen. mc 1985, Heft 5, Seite 96...99.

[2] *Thienel, Sauer, Kostal:* Entwicklungshilfe. mc 1986, Heft 10, Seite 62...64.

Wolfgang Kanis

Der Z80-EMUF

Preiswerter Einplatinen-Computer

Der mc-6504-EMUF fand eine weite Verbreitung, weil er erstens sehr preiswert ist und sich zweitens Programme für ihn mit preiswerten Tischcomputern entwickeln lassen. Beides gilt auch für den hier vorgestellten Z80-EMUF; er läßt sich mit einer Taktfrequenz bis zu 4 MHz betreiben, und die Platine kann mit 2 KByte RAM, 8 KByte EPROM und I/O-Ports mit 32 Leitungen bestückt werden.

Der Z80-EMUF (Einplatinen-Mikrocomputer für universelle Festprogramm-Anwendung) soll es allen Besitzern von Computern wie TRS-80, Video-Genie, Nascom usw. ermöglichen, ihre spezielle Computerlösung für eine bestimmte Aufgabe zu finden. Wie sein Verwandter, der 6504-EMUF [1], ist der Z80-EMUF nicht dazu gedacht, auf ihm Programme zu entwickeln (wenn das mit einem geeigneten Monitorprogramm auch prinzipiell möglich wäre), ihn groß auszubauen oder Basic-Programme laufen zu lassen. Vielmehr ist er als eine Art softwaregesteuerte Logikschaltung zu betrachten.

Maximales Z80-Minimalsystem

Diese Einschränkungen ermöglichten es, ein sehr preiswertes Minimalsystem für weniger als 100 DM zu konzipieren. Der Z80-EMUF besteht aus CPU, RAM, EPROM, PIO, Oszillator, Reset und Adressendecoder. Die genaue Verschaltung der Bauelemente ist aus *Bild 1* ersichtlich. Das Schaltbild ist für den Betrieb mit 2-KByte-Speicherbausteinen gezeichnet. Die Verwendung von 4- oder 8-KByte-Speichern ist auf der Platine schon vorgesehen. Dabei müssen die Lötbrücken entsprechend Schaltbild und Bestückungsplan (*Bild 2*) eingelötet werden. *Bild 3* und *Bild 4* zeigen das Layout der doppelseitigen Platine.

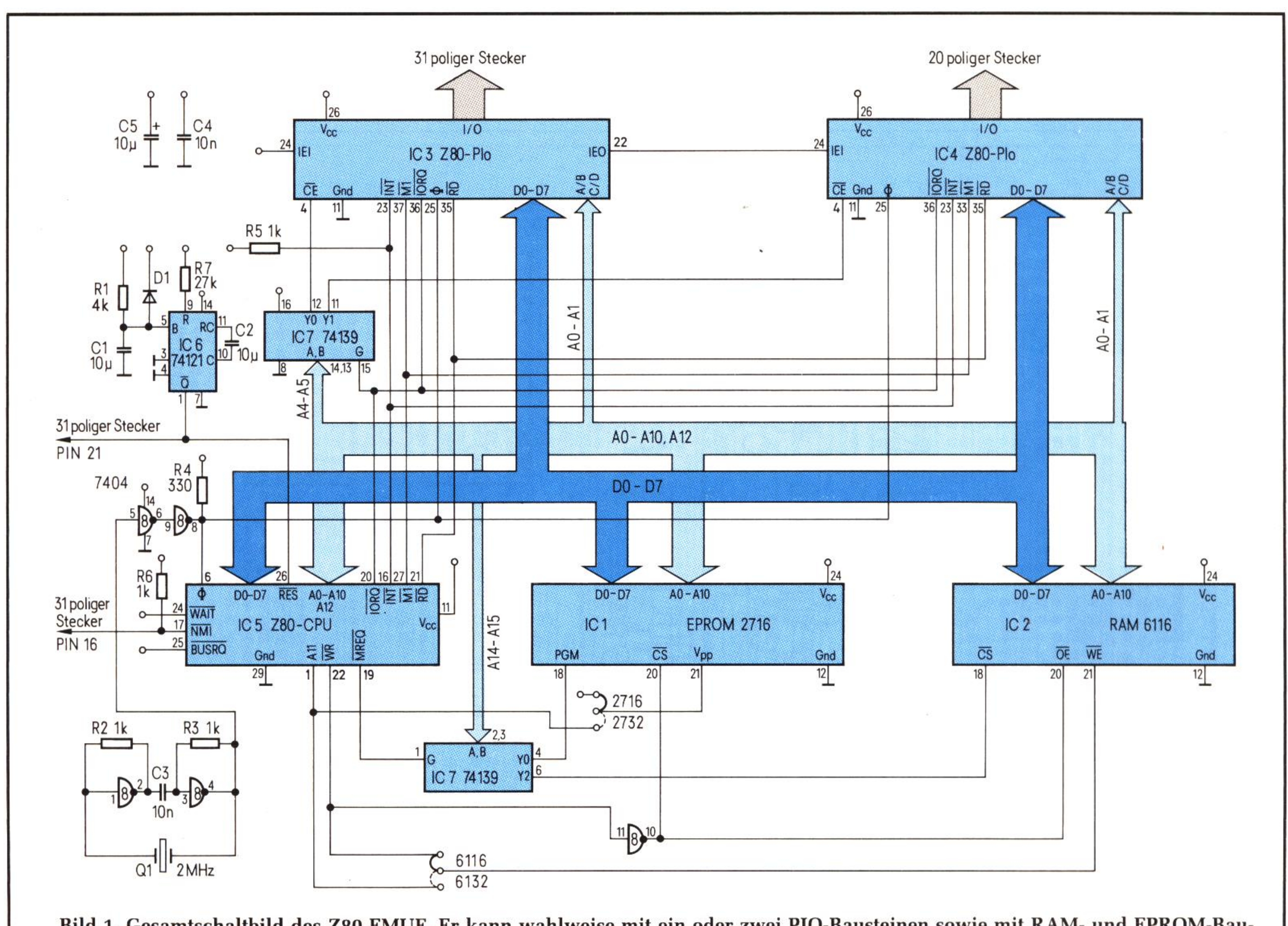


Bild 1. Gesamtschaltbild des Z80-EMUF. Er kann wahlweise mit ein oder zwei PIO-Bausteinen sowie mit RAM- und EPROM-Bausteinen unterschiedlicher Kapazität bestückt werden

Bild 3. Lötseite des Platinenlayouts

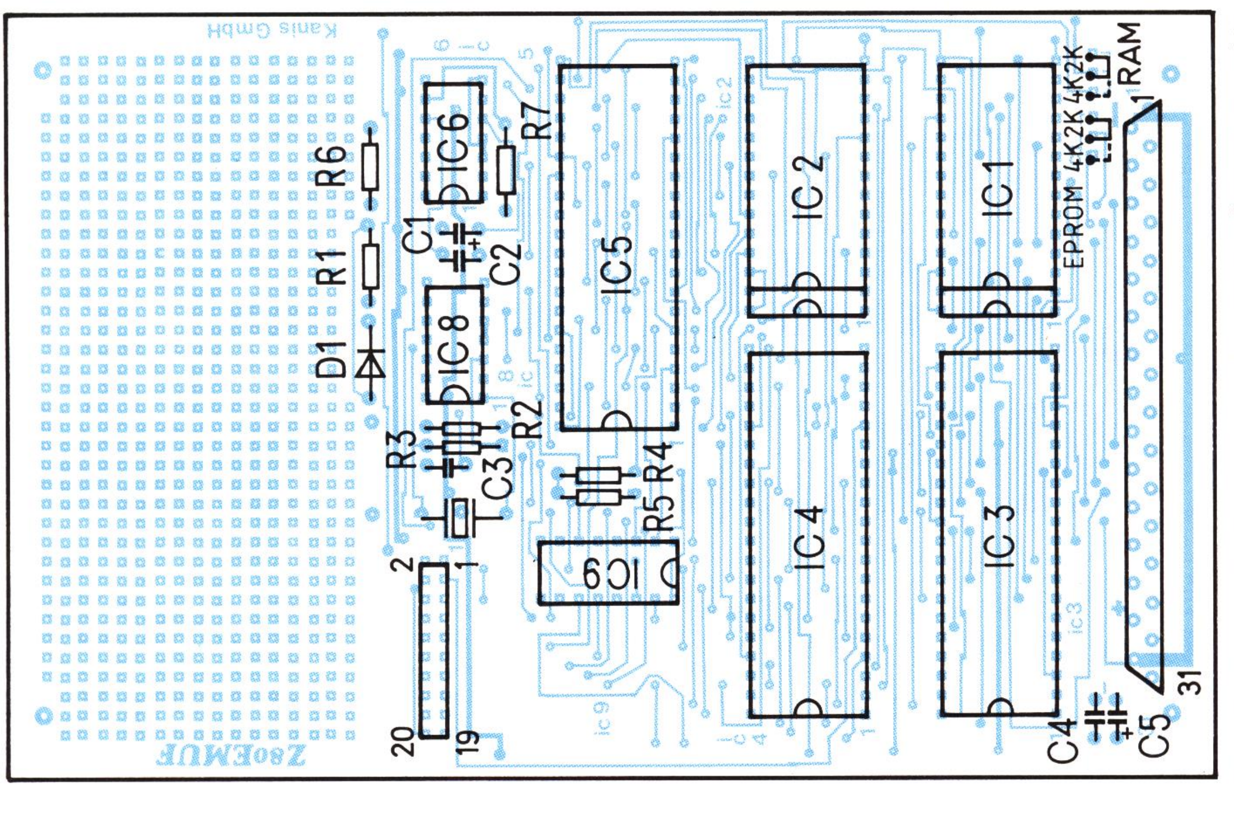


Bild 2. Bestückungsplan. Je nach verwendeter RAM- und EPROM-Kapazität sind Brücken erforderlich

```
0000                ORG 0000H
                    *Testprogramm fuer Z80 EMUF
                    *auf den Ausgaengen der PIO's
                    *wird nacheinander gezaehlt
8000                RAM EQU 8000H
0000                PIO0 EQU 0
0010                PIO1 EQU 10H
0000 3E CF          EMUT1 LD A,11001111B *Initiali-
                    *sierung aller Ports
0002 D3 02          OUT (PIO0+2),A *auf Einzelbit Ein-
                    *Ausgabe
0004 3E 00          LD A,0 *Maske alle Bits Ausgaenge
0006 D3 02          OUT (PIO0+2),A
0008 3E CF          LD A,11001111B
000A D3 03          OUT (PIO0+3),A

000C     3E 00              LD A,0
000E     D3 03              OUT (PIO0+3),A
0010     3E CF              LD A,11001111B
0012     D3 12              OUT (PIO1+2),A
0014     3E 00              LD A,0
0016     D3 12              OUT (PIO1+2),A
0018     3E CF              LD A,11001111B
001A     D3 13              OUT (PIO1+3),A
001C     3E 00              LD A,0
001E     D3 13              OUT (PIO1+3),A
0020     06 00       LOOP   LD B,0
0022     78          L1     LD A,B
0023     32 00 80           LD (RAM),A *RAM WIRD ANGESPROCHEN

0026     3A 00 80           LD A,(RAM)
0029     D3 00              OUT (PIO0+0),A
002B     05                 DEC B
002C     C2 22 00           JP NZ,L1
002F     06 00              LD B,0
0031     78          L2     LD A,B
0032     32 00 80           LD (RAM),A
0035     3A 00 80           LD A,(RAM)
0038     D3 01              OUT (PIO0+1),A
003A     05                 DEC B
003B     C2 31 00           JP NZ,L2
003E     06 00              LD B,0
0040     78          L3     LD A,B
0041     32 00 80           LD (RAM),A
0044     3A 00 80           LD A,(RAM)
0047     D3 10              OUT (PIO1+0),A

0049     05                 DEC B
004A     C2 40 00           JP NZ,L3
004D     06 00              LD B,0
004F     78          L4     LD A,B
0050     32 00 80           LD (RAM),A
0053     3A 00 80           LD A,(RAM)
0056     D3 11              OUT (PIO1+1),A
0058     05                 DEC B
0059     C2 4F 00           JP NZ,L4
005C     F2 20 00           JP LOOP
005F                        DS 1000H
105F     00                 END 2280H
```

Bild 5. Ein kleines Testprogramm für erste Versuche

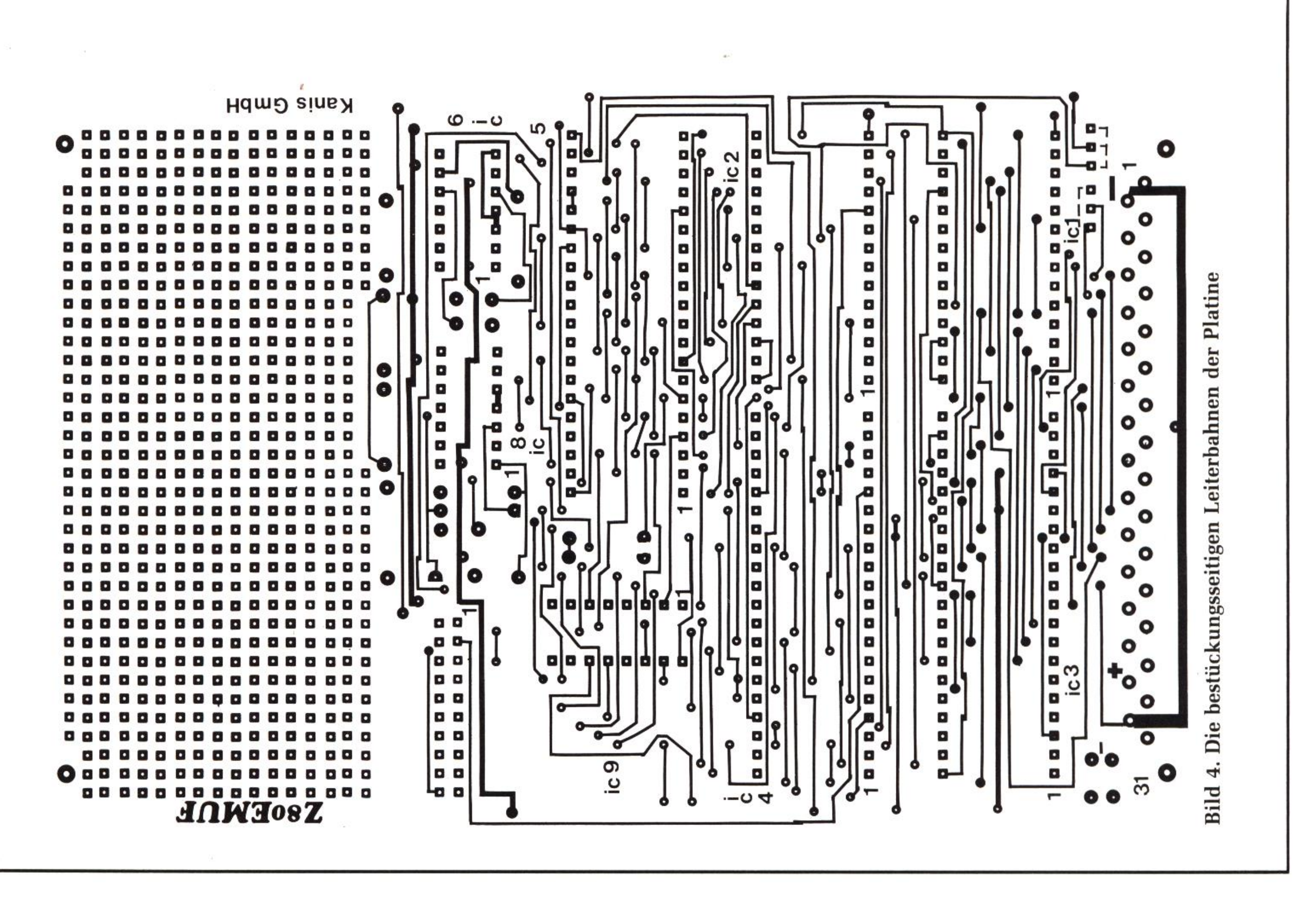


Bild 4. Die bestückungsseitigen Leiterbahnen der Platine

Das System ist über $\overline{NMI}$ und $\overline{INT}$ voll interruptfähig. PIO 0 (IC3) hat über die Daisy-Chain-Logik höchste Priorität. Wird nur eine PIO verwendet, sollte diese IC3 sein. Höhere Taktfrequenzen als 2 MHz sind bei Verwendung entsprechender Bauteile erzielbar [2, 3].

Adressenbelegung

Folgende Adressen werden durch die Adressendecoder-IC7 festgelegt:

EPROM (IC1)	ab		0000	H
RAM (IC2)	ab		8000	H
PIO0 (IC3)	A	Data	00	H
		Control	02	H
	B	Data	01	H
		Control	03	H
PIO1 (IC4)	A	Data	10	H
		Control	12	H
	B	Data	11	H
		Control	13	H

Für spezielle Anpassungen steht ein relativ großes Verdrahtungsfeld zur freien Verfügung. Auf diesem lassen sich z. B. A/D-, D/A-Wandler, Taktgenerator, Opto-Koppler o. ä. leicht aufbauen.
Um eine Verwendung der bisher erschienenen Peripherie möglichst einfach zu gestalten, wurde auf eine Kompatibilität am 31poligen Stecker mit dem 6504-EMUF geachtet. Wie aus *Tabelle 2* ersichtlich, sind alle Steckerpunkte gleich. Nur die freien Steckerpunkte wurden für zusätzliche Funktionen verwendet. Eine weitere 20polige Steckleiste dient als Anschluß für IC4 als zusätzliche PIO (*Tabelle 3*).

Die Inbetriebnahme

Um eventuelle Fehlfunktionen leichter ermitteln zu können, ist es zu empfehlen, die Platine mit Sockeln zu bestükken, mindestens für die LSI-Bauteile. Auf das richtige Einsetzen (siehe Bild 2) von IC1 und IC2 ist zu achten. Sie sollten in 28polige Sockel eingesetzt werden, dadurch ist eine spätere Erweiterung mit 4-KByte- und 8-KByte-Speicherbausteinen möglich. Vor der Inbetriebnahme ist die Platine zur Kontrolle optisch auf Lötbrücken und vergessene Lötungen abzusuchen. Nun steht einem Start nichts mehr im Wege. Das bereits programmierte EPROM einsetzen, +5 V anschließen, und schon sollte Ihr Programm abgearbeitet werden. Der Stromverbrauch beträgt in der Regel 0,3 A. Um einige Reserven zu haben, sollte ein Netzteil mit 5 V/0,5 A verwendet werden.
Sollte Ihr Programm nicht auf Anhieb laufen, empfiehlt es sich, erst den EMUF zu testen. Dies kann sehr gut mit einem kleinen Testprogramm (*Bild 5*) geschehen. Alle Ports, RAM und EPROM werden damit angesprochen. Zur Fehlersuche verwendet man am besten ein Oszilloskop. Alle Signale an EPROM, RAM, Adressendecoder und PIO werden damit untersucht. Fehlende Signale deuten auf Leiterbahnunterbrechungen, deformierte Pegel auf Kurzschlüsse hin. *Bild 6* zeigt den fertig aufgebauten Z80-EMUF.
Zum Schluß sei noch für die tatkräftige Unterstützung Herrn Rolf-Dieter Klein gedankt.

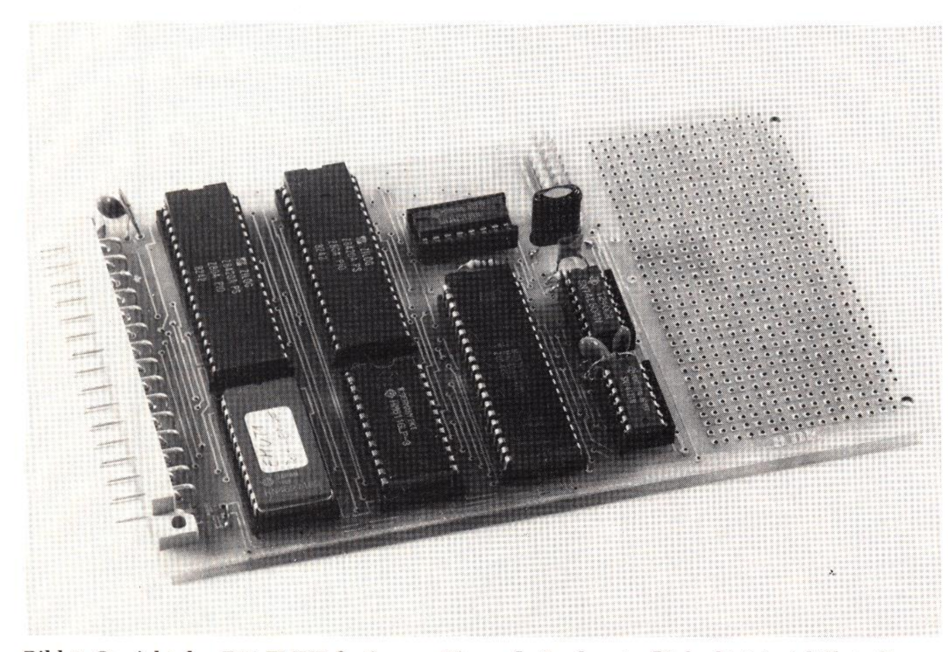


Bild 6. So sieht der Z80-EMUF fertig aus. Die große Lochrasterfläche bietet viel Platz für individuelle Erweiterungen

Platinen, Bausätze und Fertiggeräte sind beim Ing.-Büro W. Kanis, Lindenberg 113, 8134 Pöcking, erhältlich.

Literatur

[1] *Feichtinger, H.*: Mädchen für alles (6504-EMUF). mc 1981, Heft 2, und EMUF-Sonderheft, Franzis-Verlag.
[2] Zilog (Hrsg.): Z80-CPU Technical Manual.
[3] Zilog (Hrsg.): Z80-PIO Technical Manual.
[4] *Klein, M.*: Z80-Applikationsbuch, Franzis-Verlag.
[5] *Klein, R. D.*: Mikrocomputer-Hard- und Software-Praxis, Franzis-Verlag.

Tabelle 1: Stückliste

IC1 EPROM 2716 (oder entsprechende 4- bzw. 8-KByte-Speicher)
IC2 RAM 6116 (oder entsprechende 4- bzw. 8-KByte-Speicher)
IC3 PIO Z80
IC4 PIO Z80
IC5 CPU Z80
IC6 TTL 74121
IC7 TTL 74LS139
IC8 TTL 74LS04
R1 Widerstand 47 kΩ
R2 Widerstand 1 kΩ
R3 Widerstand 1 kΩ
R4 Widerstand 330 Ω
R5 Widerstand 1 kΩ
R6 Widerstand 1 kΩ
R7 Widerstand 27 kΩ
C1 Elko 1 µF
C2 Elko 10 µF
C3 Keramik-Kond. 10 nF
C4 Kermaik-Kond. 10 nF
C5 Elko 10 µF
Q1 Quarz 2 MHz (Standard)

Tabelle 2: Belegung des 31poligen Steckers (PIO – IC3)

1 – Masse	16 – $\overline{NMI}$
2 – Masse	17 – B1
3 – ARDY	18 – B2
4 – $\overline{BRDY}$	19 – B3
5 – $\overline{ASTB}$	20 – $\overline{BSTB}$
6 – A0	21 – $\overline{RES}$
7 – A1	22 – B7
8 – A2	23 – B6
9 – A7	24 – B5
10 – A6	25 – B4
11 – A5	26 – NC
12 – A4	27 – +5 V
13 – A3	28 – +5 V
14 – Masse	29 – Masse
15 – B0	30 – Masse
	31 – Masse

Tabelle 3: Belegung des 20poligen Steckers (PIO – IC4)

1 – ARDY	11 – A3
2 – $\overline{BRDY}$	12 – B3
3 – $\overline{ASTB}$	13 – A4
4 – $\overline{BSTB}$	14 – B4
5 – A0	15 – A5
6 – B0	16 – B5
7 – A1	17 – A6
8 – B1	18 – B6
9 – A2	19 – A7
10 – B2	20 – B7

Andreas Zilker, Stefan Wurdack

Z80-EMUF mit Komfort

Der Z80-EMUF ist ein kompletter Mikrocomputer und eigentlich zu schade, um nur zu Steuerungsaufgaben eingesetzt zu werden. Deshalb soll in diesem Artikel eine Möglichkeit vorgestellt werden, den EMUF mit einer einfachen Tastatur und einer Anzeige auszurüsten und ihn durch die entsprechende Software zum Leben zu erwecken. So ausgerüstet eignet er sich besonders für Entwicklungs- und Meßaufgaben oder einfach als Lehrsystem für diejenigen, die ein erstes Mal ernsthaften Kontakt zu einem Mikrocomputer suchen.

Um die CPU nicht mit der Kontrolle des Displays und der Tastatur zu belasten, wurden zwei „intelligente" Peripherietreiber verwendet. Der 7218 von Intersil steuert das 8stellige LED-Display (Typ 7218 A mit gem. Anode, B mit gemeinsamer Katode) und der 74C923 von National Semiconductor die Tastatur. Beide Bausteine werden über PIO 2 von der CPU mit Daten (PIO2A) und Steuersignalen (PIO2B) versorgt. Ein direkter Anschluß an den CPU-Bus, der prinzipiell möglich wäre, wurde nicht vorgenommen, damit die Peripherie z. B. auch für den 6504-EMUF ohne Hardwareänderungen verwendbar ist.

Der Aufbau der Hardware

Der Aufbau auf eine Einfach-Europakarte in Fädeltechnik ist problemlos möglich (*Bild 1*), wobei nicht mit Entkopplungskondensatoren gespart werden sollte (Multiplex-Störungen des Anzeigentreibers). Beide ICs, speziell der Anzeigentreiber, sind angesichts der gepflegten Preise mit großer Vorsicht zu behandeln. Für die Tastatur wurden Digitast-Mini-Keys verwendet; der 74C923 verdaut sicher auch einfachere Tasten. Außerdem ist noch Platz auf der Platine für ein Kassettenrecorder-Interface [1] und eine einfache V.24-Schnittstelle (*Bild 2*).
Der Anschluß an die CPU-Karte erfolgt mit einem 26poligen Flachbandkabel mit Pfostenstecker. Dazu wurde der „intelligenterweise" nur 20polige PIO-Anschluß der EMUF-Platine ins Lochrasterfeld herübergezogen und auf 26 Pole erweitert (zusätzlich Versorgungsspannung, RESET und NMI von der CPU). Auf keinen Fall darf man vergessen, die Handshake-Leitung STRB des A-Ports auf GND zu legen. Der EMUF selbst muß mit 4 MHz laufen, sonst stimmen die Zeitkonstanten für die serielle Schnittstelle nicht.

Schon recht komfortable Software

Das Monitor-Programm in *Bild 3* versorgt Tastatur, Anzeigen und Kassettenrecorder. Außerdem wurden einige Utilities wie z. B. MOVE und Displacement-Berechnung angefügt. Als besonderes „Zuckerl" gibt es noch einen Downloader für Host-Rechner-Kopplung und eine DCF-77-Decodierung für angeschlossenen Empfänger. Der Monitor wurde so modular wie möglich angelegt, wobei alles außer der kurzen Monitor-Hauptschleife als Unterprogramm verwendbar ist. In den Unterprogrammen werden meist nur die parametertragenden Register verändert, die zweite Registerbank der CPU bleibt frei für den Anwender. Stack und Monitor-RAM belegen die Adressen 8000H-80FFH, ab 8100H aufwärts (mögliche RAM-Erweiterung) ist Platz für „Selbstgestricktes".
Die Interrupts NMI und IRQ werden vom Monitor über RAM-Vektoren versorgt. Bei RESET wird in beiden Bereichen (NMIVE Adr. 8000H IRQVE Adr. 8003H) ein kompletter Jump-Befehl (C3...) zum Breakpoint-Entry des Monitor geladen. Die beiden Adreß-Bytes können nun vom Anwender auf seine Routinen gerichtet werden. So wird der fehlende JP (nn) des Z80 simuliert, wobei alle Register unverändert bleiben. Um die Kommando-Decodierung im Monitor für Erweiterungen zu öffnen, erfolgt am Ende der Monitor-Hauptschleife ein indirekter Sprung nach obigem Schema über den Monitor-Erweiterungsvektor (MONEVE Adr. 8006H). Er wird beim RESET auf die „Error"-Routine des Monitors gerichtet und bewirkt bei einem unbekannten Kommando eine Fehlermeldung auf dem Display. Der Benutzer kann nun diesen Vektor auf eine selbst definierte Fortsetzung des Kommando-Decoders richten und so z. B. Monitor-Erweiterungen testen, bevor sie ins EPROM kommen.

Ein kommentiertes Source-Listing ist beim Franzis-Software-Service erhältlich.

Funktionen und Kommandos

Die Belegung der Tastatur ist in *Bild 4* gezeigt, die möglichen Monitor-Kommandos sind in der *Tabelle* aufgelistet. Die meisten Kommandos sind dialogorientiert gestaltet und benutzen das 7-Segment-Display zur Textausgabe so gut es eben geht. In der Monitor-Hauptschleife stellen auch die Hex-Tasten Kommandotasten dar, zur Zifferneingabe werden sie erst innerhalb der Kommandos benutzt. Mit Hilfe der Shift-Taste (∧) stehen 38 Kommandotasten zur Verfügung. Die Verwendung der Shift-Funktion erfolgt wie beim Taschenrechner (zuerst Shift-, dann Kommandotaste). Die Shift-Funktion wird durch den

Tabelle 1: Die Monitor-Kommandos auf einen Blick

Folgende Tasten sind mit Funktionen belegt:

Taste	Funktion
0	Speicher durchblättern
1	Speicherausschnitt verschieben
2	Speicher füllen
3	Programm seriell senden bzw. auf Kassette speichern
4	Programm von Kassette bzw. serieller Schnittstelle laden
5	Verify von Kassette
6	CPU-Register nach Break anschauen evtl. ändern
7	Continue nach Break
8	Relative Sprungweite berechnen
9	DCF 77 decodieren (St. Min. Sek.)
GO	Start des USER-Programms
MEM	Memory anzeigen

Dezimalpunkt ganz rechts im Display angezeigt. Ein nicht definiertes Kommando bringt „Error" aufs Display (siehe auch Monitorerweiterung).

Bei der Hex-Eingabe werden die Ziffern von rechts nach links im Display durchgeschoben, Korrektur erfolgt durch Überschreiben. Innerhalb der einzelnen Kommandos wirkt die „+"-Taste wie eine Return-Taste zum Beenden der Ziffereingabe. Der Abschluß des Kommandos und die Rückkehr zum Monitor wird durch „Mon 3.2" angezeigt. Mittels Reset kann jedes Kommando abgebrochen werden.

Die Monitor-Kommandos im einzelnen

□ Memory-Display: Taste Mem
Hiermit können Daten im RAM abgelegt werden.

Taste	Anzeige	
Mem	M.0000	Adreßeingabe
Mem	M.aaaa dd	Dateneingabe
∧Mem	Mon	Rückkehr zur Monitor-Schleife

Mit der Mem-Taste kann zwischen Adreß- und Dateneingabe hin- und hergeschaltet werden.

□ Go-Kommando: Taste Go

Go	G 0000	Eingabe der Startadresse

Nochmaliges Betätigen der Go-Taste schickt die CPU auf die Reise. Jede andere Kommando-Taste bewirkt Rückkehr zum Monitor. Beim Start des User-Programms werden alle CPU-Register (Bank 1) aus dem Puffer im RAM geladen (siehe Break-Kommando). Ein offenes RET am Ende des Anwenderprogramms führt zurück zum Monitor.

□ Speicher durchblättern: Taste 0

0	Sta 0000	Eingabe der Startadresse
+	M.aaaa dd	
+	M.aaaa+1 dd	vorwärts blättern
Go	M.aaaa–1 dd	rückwärts blättern
Mem	M.aaaa dd	Einsprung ins normale Mem-Kommando
∧Mem	Mon	Rückkehr zum Monitor

Hier kann man z. B. nach einem bestimmten Byte im Speicher suchen, da die „+"- und die „Go"(„–")-Taste mit Autorepeat ausgestattet sind.

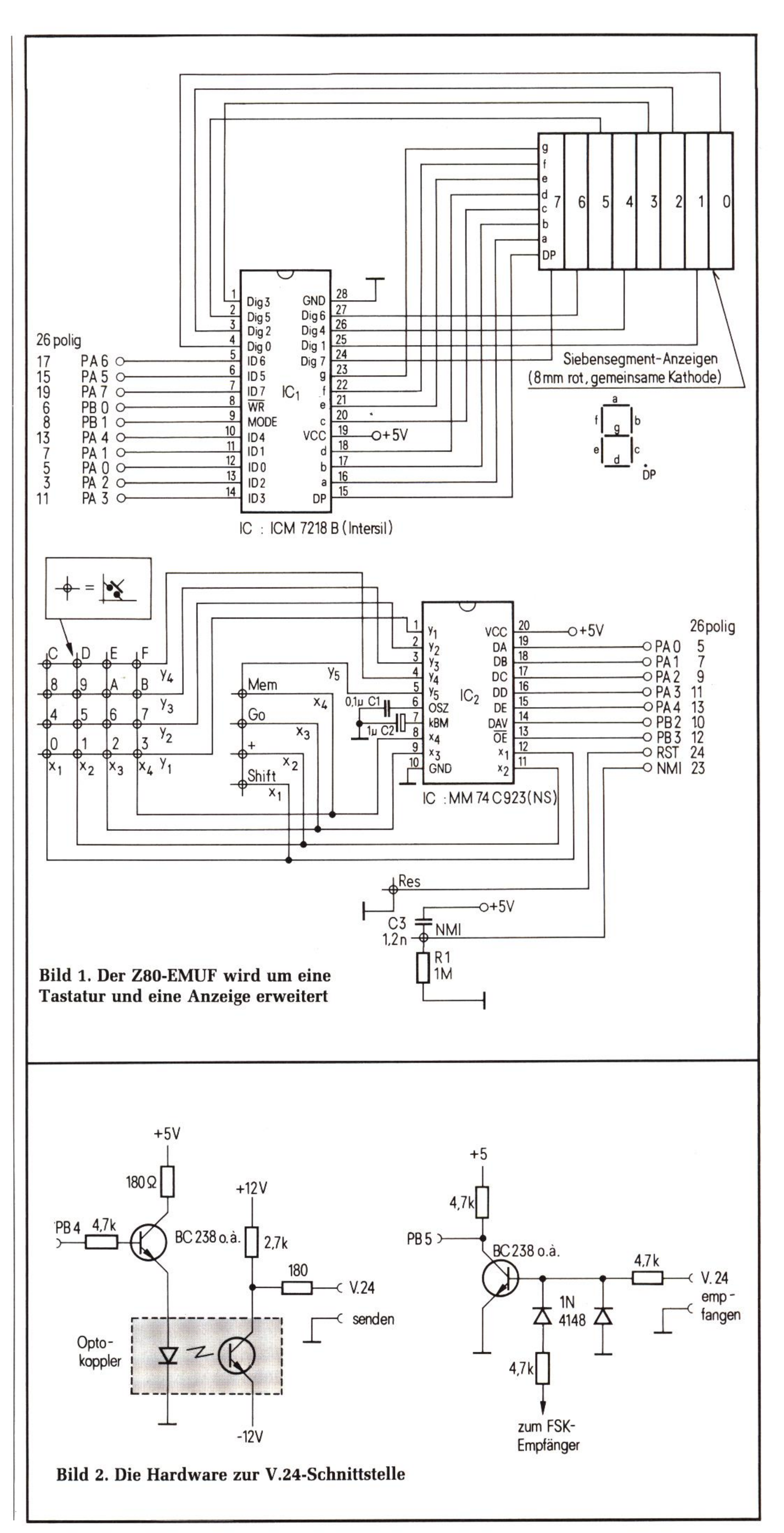


Bild 1. Der Z80-EMUF wird um eine Tastatur und eine Anzeige erweitert

Bild 2. Die Hardware zur V.24-Schnittstelle

```
5000 $0000 C3 69 00 FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  Ci..............
5010 $0010 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  ................
5020 $0020 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  ................
5030 $0030 FF FF FF FF  FF FF FF FF  C3 03 80 FB  B0 ED F5 B6  ........C..ä0muö
5040 $0040 D7 DF F0 FF  F7 FE 9F 8D  BD CF CE FF  FF FF FF FF  W_p.wß..=ON.....
5050 $0050 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  ................
5060 $0060 FF FF FF FF  FF FF C3 00  80 31 FF 80  21 FF 80 22  ......C..1..!.."
5070 $0070 25 80 21 00  00 22 09 80  22 0B 80 3E  C3 21 BC 00  %.!.."..".>C!<.
5080 $0080 32 00 80 22  01 80 32 03  80 22 04 80  21 05 07 32  2.."..2.."..!..2
5090 $0090 06 80 22 07  80 3E 7F D3  12 3E FF D3  13 3E E8 D3  ..".>.S.>.S.>hS
50A0 $00A0 13 3E 15 D3  11 3E 08 32  11 80 21 21  00 22 0F 80  .>.S.>.2..!!."..
50B0 $00B0 21 43 07 CD  11 07 CD B7  06 C3 1C 01  22 21 80 E1  !C.M..M7.C.."!.a
50C0 $00C0 22 23 80 F5  E1 22 1B 80  ED 43 1D 80  ED 53 1F 80  "#.ua"..mC..mS..
50D0 $00D0 ED 73 25 80  DD 22 27 80  FD 22 29 80  21 3B 07 CD  ms%.ü"´.ü").!;.M
50E0 $00E0 11 07 2A 23  80 22 0B 80  11 16 80 CD  C1 01 C3 1F  ..*#.".....MA.C.
50F0 $00F0 01 2A 1B 80  E5 F1 ED 4B  1D 80 ED 5B  1F 80 ED 7B  .*..eqmK..mä..mä
5100 $0100 25 80 DD 2A  27 80 FD 2A  29 80 2A 23  80 E5 2A 21  %.ü*´.ü*).*#.e*!
5110 $0110 80 C9 21 43  07 CD 11 07  CD B7 06 C9  31 00 81 21  .I!C.M..M7.I1..!
5120 $0120 1C 01 E5 CD  38 01 21 96  07 BE D2 05  07 87 5F 16  ..eM8.!..>R..._.
5130 $0130 00 23 19 5E  23 56 EB E9  C5 CD 78 01  D4 8D 01 38  .#.^#VkiEMx.T..8
5140 $0140 F8 FE 10 20  31 3A 1A 80  CB 7F 20 04  CB FF 18 02  xß. 1:..K. .K...
5150 $0150 CB BF 32 1A  80 47 CD B7  06 CD 78 01  D4 8D 01 38  K?2..GM7.Mx.T..8
5160 $0160 F8 FE 10 28  E0 CB 78 20  0D F5 CB F8  78 32 1A 80  xß.(`Kx .uKxx2..
5170 $0170 CD B7 06 F1  C6 14 C1 C9  3E 7F D3 12  DB 11 CB 5F  M7.qF.AI>.S.ä.K_
5180 $0180 20 02 37 C9  CB 97 D3 11  DB 10 E6 1F  C9 F5 DB 11   .7IK.S.ä.f.Iuä.
5190 $0190 CB 5F 20 FA  CB D7 D3 11  F1 C9 F5 D5  E5 21 3E 07  K_ zKWS.qIuUe!>.
51A0 $01A0 CD 11 07 11  14 80 2A 09  80 22 0D 80  CD 3B 02 22  M.....*.."..M;."
51B0 $01B0 09 80 FE 13  CC 5A 02 FE  27 20 E2 E1  D1 F1 C3 12  ..ß.LZ.ß´ baQqC.
51C0 $01C0 01 F5 7C CD  CF 01 7D CD  CF 01 CD B7  06 F1 C9 F5  .uöMO.üMO.M7.qIu
51D0 $01D0 F5 CB 3F CB  3F CB 3F CB  3F CD E8 01  12 13 F1 E6  uK?K?K?K?Mh...qf
51E0 $01E0 0F CD E8 01  12 13 F1 C9  E5 21 3B 00  85 6F 7E E1  .Mh...qIe!;..oßa
51F0 $01F0 C9 F5 D5 E5  11 19 80 2A  09 80 7E CD  CF 01 CD B7  IuUe...*..ßMO.M7
5200 $0200 06 E1 D1 F1  C9 D5 E5 2A  09 80 23 22  09 80 11 14  .aQqIUe*..#"....
5210 $0210 80 CD C1 01  E1 D1 C9 D5  E5 2A 09 80  2B 22 09 80  .MA.aQIUe*..+"..
5220 $0220 11 14 80 CD  C1 01 E1 D1  C9 D5 CD 11  07 11 17 80  ...MA.aQIUM.....
5230 $0230 CD 35 02 D1  C9 21 00 00  22 0D 80 D5  D1 D5 2A 0D  M5.QI!.."..UQU*.
5240 $0240 80 CD C1 01  CD 38 01 FE  10 30 0A 21  0D 80 ED 6F  .MA.M8.ß.0.!..mo
5250 $0250 23 ED 6F 18  E7 2A 0D 80  D1 C9 D5 E5  CD F1 01 CD  #mo.g*..QIUeMq.M
5260 $0260 38 01 FE 13  28 19 FE 27  28 15 FE 11  CC 05 02 FE  8.ß.(.ß´(.ß.L..ß
5270 $0270 12 CC 17 02  FE 10 30 E4  2A 09 80 ED  6F 18 DD E1  .L..ß.0d*..mo.üa
5280 $0280 D1 C9 21 40  07 CD 11 07  11 16 80 2A  0B 80 22 0D  QI!§.M.....*..".
5290 $0290 80 CD 3B 02  22 0B 80 FE  12 28 03 C3  12 01 FD 2A  .M;."..ß.(.C..ü*
52A0 $02A0 29 80 DD 2A  27 80 ED 5B  1F 80 ED 4B  1D 80 2A 1B  ).ü*´.mä..mK..*.
52B0 $02B0 80 E5 F1 ED  7B 25 80 2A  0B 80 E5 2A  21 80 C9 21  .eqmä%.*..e*!.I!
52C0 $02C0 4B 07 CD 29  02 22 09 80  21 3E 07 CD  11 07 2A 09  K.M)."..!>.M..*.
52D0 $02D0 80 11 14 80  CD C1 01 CD  F1 01 CD 78  01 38 FB FE  ....MA.Mq.Mx.8äß
52E0 $02E0 13 CA 9A 01  FE 12 28 04  FE 11 20 EE  21 D0 07 CD  .J..ß.(.ß. n!P.M
52F0 $02F0 A7 06 FE 12  28 05 CD 05  02 18 DC CD  17 02 18 D7  ´.ß.(.M...öM...W
5300 $0300 21 4B 07 CD  29 02 E5 21  4F 07 CD 29  02 D1 D5 AF  !K.M).e!O.M).QU/
5310 $0310 ED 52 FA 05  07 23 E5 21  53 07 CD 29  02 E5 D1 C1  mRz..#e!S.M).eQA
5320 $0320 E1 D5 E5 AF  ED 52 E1 D1  38 04 ED B0  18 08 09 2B  aUe/mRaQ8.m0...+
5330 $0330 EB 09 2B EB  ED B8 C3 12  01 21 79 07  CD 29 02 4D  k.+km8C..!y.M).M
5340 $0340 21 4B 07 CD  29 02 E5 21  4F 07 CD 29  02 D1 AF ED  !K.M).e!O.M).Q/m
5350 $0350 52 DA 05 07  23 EB 71 23  1B 7A B3 20  F9 C3 12 01  RZ..#kq#.z3 yC..
5360 $0360 21 56 07 CD  29 02 E5 21  53 07 CD 29  02 EB E1 23  !V.M).e!S.M).ka#
5370 $0370 23 EB AF ED  52 FA 83 03  7C FE 00 20  1F CB 7D 20  #k/mRz..öß. .Kü 
5380 $0380 1B 18 09 7C  FE FF 20 14  CB 7D 28 10  7D 21 5A 07  ...öß. .Kü(.ü!Z.
5390 $0390 CD 11 07 11  19 80 CD CF  01 C3 B7 06  21 5F 07 CD  M.....MO.C7.!_.M
53A0 $03A0 11 07 C3 B7  06 06 08 11  1B 80 21 7E  07 CD 11 07  ..C7......!ß.M..
53B0 $03B0 E5 1A 32 0D  80 13 1A 32  0E 80 1B D5  11 17 80 CD  e.2....2...U...M
53C0 $03C0 3B 02 D1 7D  12 13 7C 12  13 E1 10 E1  C3 12 01 21  ;.Qü..ö..a.aC..!
53D0 $03D0 4B 07 CD 29  02 E5 21 4F  07 CD 29 02  D1 E5 AF ED  K.M).e!O.M).Qe/m
53E0 $03E0 52 E1 DA 05  07 CA 05 07  E5 21 70 07  CD 29 02 C1  RaZ..J..e!p.M).A
53F0 $03F0 E5 C5 21 74  07 CD 29 02  E5 21 68 07  CD 11 07 CD  eE!t.M).e!h.M..M
5400 $0400 B7 06 E1 AF  CD 20 05 3E  A5 CD 20 05  3E 55 CD 20  7.a/M .>%M .>UM 
5410 $0410 05 7D CD 20  05 7C CD 20  05 E1 E5 23  AF ED 52 28  .üM .öM .ae#/mR(
5420 $0420 02 30 12 E1  E1 3E 78 CD  20 05 7D CD  20 05 7C CD  .0.aa>xM .üM .öM
5430 $0430 20 05 C3 12  01 01 00 01  E5 AF ED 42  E1 38 02 2E   .C.....e/mBa8..
5440 $0440 00 45 3E 3C  CD 20 05 78  CD 20 05 7B  CD 20 05 7A  .E><M .xM .äM .z
5450 $0450 CD 20 05 83  4F 1A 81 4F  1A CD 20 05  D5 11 19 80  M ..O..O.M .U...
5460 $0460 CD CF 01 CD  B7 06 D1 13  10 EB 79 CD  20 05 18 A9  MO.M7.Q..kyM ..)
5470 $0470 21 00 00 3A  12 80 CB C7  32 12 80 18  0E 3A 12 80  !..:..KG2....:..
5480 $0480 CB 87 32 12  80 21 74 07  CD 29 02 E5  21 6C 07 CD  K.2..!t.M).e!l.M
5490 $0490 11 07 11 19  80 3E 00 CD  CF 01 CD B7  06 E1 CD 4E  .....>.MO.M7.aMN
54A0 $04A0 05 FE A5 20  F9 CD 4E 05  FE 55 20 F2  CD 4E 05 11  .ß% yMN.ßU rMN..
54B0 $04B0 13 80 4F CD  4E 05 47 CD  CF 01 79 CD  CF 01 CD B7  ..OMN.GMO.yMO.M7
54C0 $04C0 06 7C B5 28  08 79 BD 20  D5 78 BC 20  D1 CD 4E 05  .ö5(.y= Ux< QMN.
54D0 $04D0 FE 78 20 0D  CD 4E 05 6F  CD 4E 05 67  B5 CA 12 01  ßx .MN.oMN.g5J..
54E0 $04E0 E9 FE 3C 20  E8 CD 4E 05  47 CD 4E 05  6F CD 4E 05  iß< hMN.GMN.oMN.
54F0 $04F0 67 85 4F CD  4E 05 F5 3A  12 80 CB 47  28 07 F1 BE  g.OMN.u:..KG(.q>
5500 $0500 C2 05 07 18  02 F1 77 23  D5 11 19 80  CD CF 01 CD  B....qw#U...MO.M
5510 $0510 B7 06 D1 81  4F 10 DC CD  4E 05 B9 28  B0 C3 05 07  7.Q.O.öMN.9(0C..
5520 $0520 F5 E5 D5 57  2A 0F 80 3A  11 80 5F 1C  DB 11 CB A7  ueUW*..:.._.ä.K´
5530 $0530 D3 11 E5 CD  A7 06 E1 CB  3A CB A7 30  02 CB E7 1D  S.eM´.aK:K´0.Kg.
5540 $0540 20 EE 29 CB  E7 D3 11 CD  A7 06 D1 E1  F1 C9 E5 D5   n)KgS.M´.QaqIeU
5550 $0550 2A 0F 80 3A  11 80 5F 16  00 CB 3C CB  1D DB 11 CB  *..:.._...K<K.ä.K
5560 $0560 6F 20 FA E5  CD A7 06 E1  DB 11 CB 6F  20 EF 2A 0F  o zeM´.aä.Ko o*.
5570 $0570 80 E5 CD A7  06 E1 DB 11  37 CB 6F 20  01 3F CB 1A  .eM´.aä.7Ko .?K.
5580 $0580 E5 CD A7 06  E1 1D 20 EE  3A 11 80 5F  3E 08 93 28  eM´.a. n:.._>..(
5590 $0590 05 CB 3A 3D  18 F9 7A D1  E1 C9 21 7D  07 CD 11 07  .K:=.yzQaI!ü.M..
55A0 $05A0 CD B7 06 AF  32 2B 80 32  2C 80 32 2E  80 CD 9A 06  M7./2+.2,.2..M..
55B0 $05B0 06 15 CD 74  06 CD 0F 06  CD 1C 06 10  F5 06 08 CD  ..Mt.M..M...u..M
55C0 $05C0 4E 06 20 D6  79 CB BF 32  2D 80 06 06  CD 4E 06 20  N. VyK?2-...MN. 
55D0 $05D0 C9 79 32 2F  80 06 17 CD  74 06 CD 0F  06 CD 1C 06  Iy2/...Mt.M..M..
55E0 $05E0 10 F5 CD 74  06 21 10 27  CD A7 06 CD  74 06 3A 2D  .uMt.!.´M´.Mt.:-
55F0 $05F0 80 32 2C 80  3A 2F 80 32  2E 80 AF 32  2B 80 21 64  .2,.:/.2../2+.!d
5600 $0600 00 CD A7 06  DB 11 CB 77  28 90 CD 1C  06 18 A1 21  .M´.ä.Kw(.M...!!
5610 $0610 00 00 DB 11  CB 77 C0 CD  29 06 18 F6  21 00 00 DB  ..ä.Kw§M)..v!..ä
5620 $0620 11 CB 77 C8  CD 29 06 18  F6 E5 21 0A  00 CD A7 06  .KwHM)..ve!..M´.
5630 $0630 E1 23 C9 AF  B4 20 16 7D  FE 78 30 07  FE 50 38 0D  a#I/4 .üßx0.ßP8.
5640 $0640 AF 18 0A FE  DC 30 06 FE  B4 38 02 AF  37 C9 C5 AF  /..ßö0.ß48./7IE/
5650 $0650 4F CD 74 06  CD 0F 06 CD  33 06 CB 19  B7 20 13 CD  OMt.M..M3.K.7 .M
5660 $0660 1C 06 10 ED  79 C1 4F 3E  08 90 C8 47  CB 19 10 FC  ...myAO>..HGK..ö
5670 $0670 AF C9 C1 C9  11 19 80 3A  2B 80 F5 CD  CF 01 11 16  /IAI...:+.uMO...
5680 $0680 80 3A 2C 80  CD CF 01 11  13 80 3A 2E  80 CD CF 01  .:,.MO....:..MO.
5690 $0690 CD B7 06 F1  3C 27 32 2B  80 C9 CD 1C  06 7C FE 08  M7.q<´2+.IM..öß.
56A0 $06A0 30 F8 FE 05  38 F4 C9 F5  3E 1A 3D C2  AA 06 C0 2B  0xß.8tIu>.=B*.§+
56B0 $06B0 7C B5 C2 A8  06 F1 C9 F5  C5 E5 3E 3F  D3 12 3E F0  ö5B(.qIuEe>?S.>p
56C0 $06C0 D3 10 DB 11  CB CF D3 11  CB 87 D3 11  CB C7 D3 11  S.ä.KOS.K.S.KGS.
56D0 $06D0 CB 8F D3 11  06 08 21 1A  80 7E D3 10  DB 11 CB 87  K.S...!..ßS.ä.K.
56E0 $06E0 D3 11 CB C7  D3 11 2B 10  F0 3E F0 CB  BF D3 10 DB  S.KGS.+.p>pK?S.ä
56F0 $06F0 11 CB CF D3  11 CB 87 D3  11 CB C7 D3  11 CB 8F D3  .KOS.K.S.KGS.K.S
5700 $0700 11 E1 C1 F1  C9 E5 21 34  07 CD 11 07  CD B7 06 E1  .aAqIe!4.M..M7.a
5710 $0710 C9 F5 C5 D5  11 13 80 7E  23 47 4F B7  28 06 7E 12  IuEU...ß#GO7(.ß.
5720 $0720 23 13 10 FA  3E 08 91 28  07 47 3E 80  12 13 10 FC  #..z>..(.G>....ö
5730 $0730 D1 C1 F1 C9  06 80 CF 8C  8C 9D 8C 02  9F 8C 01 7A  QAqI..O........z
5740 $0740 02 DB 9D 07  80 FA 9D 9C  80 75 ED 03  D7 8F FE 03  .ä...z...um.W.ß.
5750 $0750 CF 9C BD 02  8F 9D 03 9F  8C FE 04 BD  B0 D7 EE 08  O.=......ß.=0Wn.
5760 $0760 8F 9D 9D 80  8B 9D 9C DB  03 D7 CF 9C  03 CF FA EE  .......ä.WO..Ozn
5770 $0770 03 CF D7 EE  04 B0 BD 9C  0C 03 CE B0  8B 00 02 FE  .OWn.0=...N0...ß
5780 $0780 CE 02 9F CB  02 BD CF 02  BE 8B 02 EE  CB 02 D7 EE  N..K.=O.>..nK.Wn
5790 $0790 02 B0 BE 02  B0 B7 14 BF  02 00 03 39  03 CF 03 7D  .0>.07.?...9.O.ü
57A0 $07A0 04 70 04 A5  03 F1 00 60  03 9A 05 06  80 06 80 06  .p.%.q.`........
57B0 $07B0 80 06 80 06  80 06 80 06  80 06 80 82  02 9A 01 FF  ................
```

Bild 3. Das Monitorprogramm erweckt den Z80-EMUF zum Leben. Es ist vom Franzis-Software-Service auch im EPROM erhältlich

□ MOVE-Kommando: Taste 1

1	Sta	0000	Startadresse des Blocks eingeben
+	End	0000	Endadresse
+	to	0000	Zieladresse
+	Mon		

Bei Startadresse ≦ Endadresse erfolgt eine Fehlermeldung

□ FILL-Kommando: Taste 2

2	FIL	0000	Eingabe des „Fill-Bytes" (nur die beiden rechten Ziffern sind gültig)
+	Sta	0000	Startadresse
+	End	0000	Endadresse
+	Mon		

□ SAVE-Kommando: Taste 3

3	Sta	0000	Start des Speicherblocks
+	End	0000	Ende des Speicherblocks
+	Esp	0000	Einsprungadresse (siehe unten)
+	ldnr	0000	Eingabe einer vierstelligen Kennummer des Programmblocks auf Band
+	Sen	XX	XX „flimmernde" Sendedaten

Die Einsprungadresse wird mit aufgezeichnet und nach dem Laden des Datenblocks startet der Prozessor an dieser Stelle. Dadurch wird ein Autostart des Programms möglich. Wird als Einsprungadresse 0000 eingegeben bzw. „+" gedrückt, dann kehrt der Rechner nach dem Laden zum Monitor zurück.

□ LOAD-Kommando: Taste 4

4	Idnr	0000	Eingabe der Kennummer des zu ladenden Programms. Wird 0000 eingegeben bzw. „+" gedrückt, dann wird das nächste vollständige Programm vom Band geladen.
+	Emp	00	Empfangsbereitschaft des Prozessors

Sobald Daten empfangen werden, verändert sich die Anzeige:

iiii	xx	iiii Id-Nummer des empfangenen Programms xx empfangene Daten

nach abgeschlossenem Ladevorgang siehe LOAD-Kommando.

□ Verify-Kommando: Taste 5

5	EMP	00	siehe LOAD

Das auf Band geschriebene Programm kann mit dem Speicherinhalt verglichen werden. Wenn keine Fehler aufgetreten sind, erfolgt Rückkehr zum Monitor, andernfalls wird „Error" gemeldet und die Operation abgebrochen.

□ Anzeige der CPU-Register: Taste 6

Diese Funktion ist nur sinnvoll nach einem NMI oder Breakpoint. Nur dann sind die Puffer-Zellen im RAM auf aktuellem Stand.

6	AF	aaff	Anzeige von Akku und Flags, zu ändern wie Memory-Zellen
+	BC	bbcc	
...			
+	IY	yyyy	
+	Mon		

□ Continue-Kommando: Taste 7

Auch dieses Kommando sollte nur nach NMI oder Break verwendet werden. Andernfalls kann es zum Absturz des Rechners führen.
Die eventuell modifizierten RAM-Zellen werden in die CPU-Register geladen und das Programm beim augenblicklichen Programmzählerstand fortgesetzt.

□ Displacement-Rechner: Taste 8

Hiermit kann der Offset für die relativen Sprünge der Z80-CPU berechnet werden.

8	bra	0000	Adresse des JR-Befehls (nicht des Displacement-Bytes) eingeben
+	to	0000	Adresse des Ziel-Labels
+	disp	xx	Displacement

Wurde die maximale Sprungweite überschritten, so erscheint die Meldung „too long" im Display.

□ DCF-77-Decodierung: Taste 9

Mit diesem Kommando kann der „Kleine" nach Anschluß eines einfachen DCF-77-Empfängers an PIO2, PB 6, in eine hochgenaue Uhr verwandelt werden. Der Empfänger sollte TTL-Pegel (Ruhezustand high) liefern. Der EMUF verdaut für log. 0 80...120 ms Absenkung, für log. 1 180...220 ms.

Um den Programmaufwand gering zu halten, decodiert der Rechner nur Stunden, Minuten und Sekunden und zeigt sie auf dem Display an. Nach Start des Programms wartet der EMUF die 59-Sekunden-Marke ab und zeigt 00 00 an. Nach einer weiteren Minute werden auch die Stunden und Minuten angezeigt, die maximale Synchronisationszeit beträgt also 1 Minute und 59 Sekunden. Treten während des Empfangs Störungen auf, so erlischt das Display, um sinnlose Zeitanzeigen zu vermeiden; der Rechner versucht erneut zu synchronisieren. Aus diesem Programm kann der EMUF nur mit Reset zurückgeholt werden.

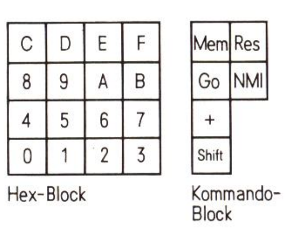


Bild 4. Die Belegung der kleinen Tastatur

BREAK- und STOP-Befehl

Als BREAK-Befehl wird, wie allgemein üblich, der RST 38H (0FFH) verwendet. Taucht er im Programm auf, so werden die CPU-Register ins RAM gerettet und man landet im Monitor (siehe auch Kommando 6 und 7).

Der NMI der CPU wird als STOP-Befehl interpretiert, um die CPU definiert aus endlosen Schleifen herauszuholen.
Die Register werden ebenfalls gerettet. Dieses Kommando funktioniert nur, solange der NMI-Vektor im RAM nicht verändert wird. Zum Entprellen der NMI-Taste wird ein einfaches RC-Glied verwendet (siehe Bild 1).

Im folgenden soll noch ausführlich auf die Möglichkeiten des seriellen Empfangs bzw. Sendens mittels des softwaremäßig implementierten USART's eingegangen werden.

Die Tasten 3 + 4 zum Abspeichern und Laden von Programmen bedienen sich des Datenformats in *Tabelle 2*.

Der Markierzustand der Schnittstelle ist high. Die Daten haben seriell folgende Form:

8 Bit breite Worte, 1 Startbit, 2 Stopbits. Die Übertragungsrate beträgt 300 Baud. Durch Änderung der entsprechenden RAM-Adressen (BITNR 8011H, BAUDCT 800FH) lassen sich Baudraten bis zu 2400 Baud und beliebige (bis 8 Bit) Wortbreiten einstellen. Entsprechende Versuche sind bereits erfolgreich durchgeführt worden. Die Sendung von

einem Startbit und 2 Stopbits ist fest programmiert.

Hierdurch ergibt sich die interessante Möglichkeit, Programme, die dieses Format einhalten, von einem Host-Rechner zu laden und automatisch ausführen zu lassen. Wer dieses Schema nicht einhalten möchte, kann direkt auf die Schnittstellentreiber zugreifen und sein eigenes Übertragungsprotokoll ablaufen lassen.

Der Rechner könnte durch Nutzung der seriellen Schnittstelle und entsprechender Programme völlig ferngesteuert werden. Außerdem gilt es noch zu beachten, daß er nur entweder senden oder empfangen kann. Beides gleichzeitig ist nicht möglich.

Die Aufzeichnung der Programme mittels des (Funkschau-)FSK-Modems funktionierte auf Anhieb. Die Anforderungen in bezug auf Kassetten- und Recorderqualität sind gering. Da keine Umschaltung von Recorderinterface auf die serielle Schnittstelle notwendig ist, versteht es sich von selbst, daß beim Empfang nur eine der beiden Quellen in Betrieb sein darf. Die Anschlüsse der nicht benutzten Schnittstelle müssen offen bleiben.

Tabelle 2: Das Datenformat beim Speichern auf Kassette

gesendete Bytes	Anzahl	
0	1	
A5H	1	Marken für Programmbeginn
55H	1	
Kennummer	2	Reihenfolge low Byte, high Byte
3CH	1	Startbyte für 1. Datenfeld
Länge	1	Länge des Datenfeldes, max. 256 Bytes/Block
Quelladresse	2	low Byte, high Byte
Datenfeld	Länge	
Checksum	1	(Summe aller gesendeten Bytes pro Block) modulo 256
2. Datenblock		
n-ter Datenblock		
letzter Datenblock		
78H	1	Marke Fileende
Einsprungadresse	2	low Byte, high Byte

Wichtige RAM-Adressen

8000H NMIVE

Sprungvektor für NMI 'C3 xx xx'

8003H IRQVE

Sprungvektor für Interrupt Mode 1 bzw. RST 38H.
Analog NMIVE

8006H MONEVE

Monitorerweiterungsvektor. Zeigt nach der Initialisierung auf die Error-Routine. Hier kann der Benutzer Fehlermeldungen abfangen, die nach einer nicht definierten Taste in der Monitorschleife ausgegeben werden. Der Akku enthält zu diesem Zeitpunkt 'Tastencode * 2'. 'SLR A' ergibt den Tastencode.

800FH BAUDCT

Zeitkonstante für Baudrate. Durch entsprechende Werte lassen sich die Baudraten einstellen. Folgende Beziehung gilt:

1/Baudrate = (BAUDCT)*100 Mikrosekunden

Folgende Baudraten wurden erprobt:

Baudrate	BAUDCT
75	83H
150	42H
300	21H
600	10H
1200	8H
2400	4H

Nach einem Reset sind 300 Baud eingestellt.

8011H BITNR

Anzahl der Bits pro seriellem Datenwort.

8013H DISBUF

Displaypuffer. Hier stehen die Bitmuster für den aktuellen Displayinhalt (8 Zeichen).

Segment – Bit Zuordnung (s. Bild 1):

Bit	7	6	5	4	3	2	1	0
Segment	DP	a	b	c	e	g	f	d

Wichtige ROM-Adressen und Unterprogramme

011FH		Monitorwarmstart
0796H		Kommandotabelle
0138H	KEYIN	Wartet auf Taste. Tastenwert im Akku, berücksichtigt Shift-Taste
0178H	GETKEY	Tastenfeldabfrage Taste gedrückt → Carry = 0, A = Tastencode Taste nicht gedrückt → Carry = 1, A = xx
01C1H	HLDISP	HL als 4-Digit-Zahl ausgeben. DE ist Pointer auf 1. Digit
01CFH	TODIBUF	Hexzahl im Akku wird als Bitmuster im Displaypuffer abgelegt. DE ist Pointer auf 1. Stelle
0235H	HLINCL	Eingabe einer 4stelligen Hexzahl in HL. Durchschieben und Echo auf Display (DE als Pointer). Abbruch der Eingabe durch Kommandotaste (> 0FH)
05200	SERSEN	Byte in A seriell senden (siehe BAUDCT und BITNR)
054E	SEREMP	Byte seriell empfangen (siehe oben). Empfangenes Byte im Akku
06B7H	DISPOUT	Ausgabe des Displaypuffers aufs Display (siehe DISBUF)
0711H	FILBUF	Zeichenkette in den Displaypuffer schreiben; Format: n CC...C $0 \leqq n \leqq 8$ n = Anzahl der Textzeichen (Bitmuster) Register HL zeigt auf „n"

Literatur

[1] *Feichtinger, Herwig:* FSK-Demodulator/-Oszillator. Funkschau 1978, Heft 14, Seite 698 und Heft 15, Seite 742.

Dr. Dieter Götz

Z80-EMUF mit Display und Tastatur

In mc 1983, Heft 4, wurde der äußerst preiswerte Z80-EMUF vorgestellt [1]. Im vollausgebauten Zustand verfügt er über 40 programmierbare Ein-/Ausgabeleitungen und zusätzlich über eine Reihe von Steuer- und Interruptleitungen. Ziel dieses Beitrages ist es, seinen Einsatz durch den Anschluß einer einfachen Tastatur und Anzeige sowie durch einen entsprechenden Monitor noch vielseitiger zu machen.

Das Grundkonzept war, mit möglichst wenig Hardware auszukommen, andererseits aber die Zahl der vorhandenen Ein-/Ausgabeleitungen für den späteren Anschluß von A/D- bzw. D/A-Wandlern zu erhalten.
Es wurde deshalb eine zusätzliche Schnittstelle, und zwar der Parallelbaustein 8255, an den EMUF angeschlossen. Er findet neben der Stromversorgung auf dem reichlich bemessenen Lochrasterteil der EMUF-Platine Platz. Zur Adreßdecodierung wird ein noch freier Anschluß des 2-Bit-Binärdecoders (74139) des EMUF verwandt (Pin 10). Die Adressen für den 8255 sind 20H...23H. Außer der Schnittstelle werden nur noch sechs 7-Segment-Anzeigen, 13 Transistoren, 16 Tasten und einige Widerstände benötigt. Das Multiplexen der sechs Digits und die Codierung der 7-Segment-Ziffern wird dem EMUF übertragen.

Von den Ein-/Ausgabeleitungen des EMUF wird nur eine einzige benötigt und zwar Bit 7 von Port B des Z80-PIO 0. Sie dient zum seriellen Datenaustausch mit anderen Computern. Im vorliegenden Fall war dies ein TRS-80, M1, auf dem z. B. größere Programme erstellt und getestet wurden, die dann anschließend zum EMUF transferiert wurden. Andererseits können auch vom EMUF gesammelte Daten zur Weiterverarbeitung, z. B. für eine graphische Auswertung, überspielt werden.

Bild 1. So wird der Portbaustein 8255 an die Z80-CPU angeschlossen

Der Anschluß der Siebensegment-Anzeigen

Bild 1 zeigt den Anschluß der Schnittstelle an die Z80-CPU. Das Reset-Signal des Z80 muß noch invertiert werden. Dies geschieht über die noch freien Pins 13 und 12 des sich auf der Platine befindenden Inverterbausteins. Von Port A des 8255 werden die Ausgänge 0...5, von Port B die Ausgänge 0 und 2...7 benötigt. Die Tastatur wird an Port C angeschlossen.

Da die Stromaufnahme der verwendeten Siebensegment-Anzeigen (HA 1077y von Siemens, gemeinsame Kathode) relativ gering ist, genügen als Treiber Transistoren vom Typ BC107. Der Anschluß der Transistoren als Segment- bzw. Zifferntreiber ist aus *Bild 2* ersichtlich.

Die Zuordnung der Matrixelemente der Tastatur

Matrix-element	Bedeutung ohne Shift	Bedeutung mit Shift
07	Ziffer: 0	Ziffer: 8
17	1	9
27	2	A
37	3	B
06	4	C
16	5	D
26	6	E
36	7	F
04	Shift (S)	Shift (S)
14	Clear (CR)	Out (O)
24	Break (BK)	In (IN)
34	Display (DP)	–
05	Enter (EN)	–
15	Insert (I)	–
25	Breakpoint (BP)	–
35	Go (G)	–

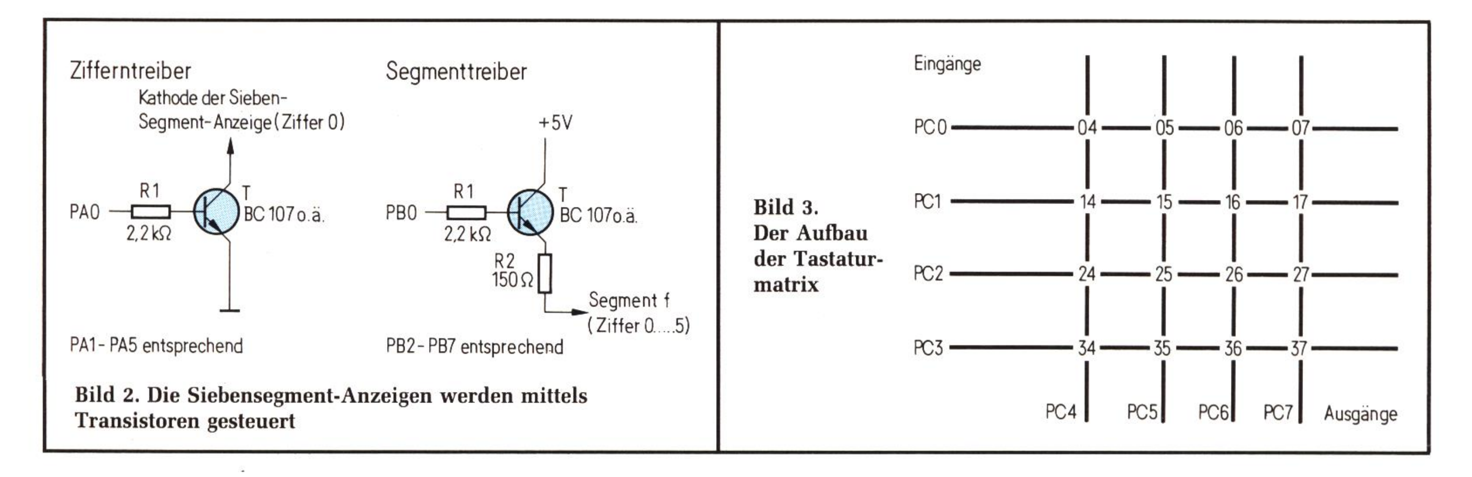


Bild 2. Die Siebensegment-Anzeigen werden mittels Transistoren gesteuert

Bild 3. Der Aufbau der Tastaturmatrix

Zum Eingeben braucht man eine Tastatur

Auch bei der Tastatur wurde das eingangs erwähnte Konzept einer Minimierung der Hardware eingehalten. Es wurden lediglich 16 Tasten verwendet. Um neben den 16 Ziffern noch zusätzliche Funktionstasten zu haben, wurden die Tasten zum größten Teil doppelt belegt. *Bild 3* zeigt den Anschluß der Tastaturmatrix. Die Bedeutung der Matrixelemente zeigt die *Tabelle*.

Der Monitor mit seinen Funktionen

Das Listing des Monitors ist aus *Bild 4* ersichtlich.
Beim Einschalten des EMUF meldet sich der Monitor mit 'r u n'. Mit Ausnahme des IN-, O- und CR-Befehls ist jetzt eine entsprechende Speicherstelle einzugeben (zur Erinnerung: ROM:0000-07FF; RAM:8000-87FF). Die Speicherplätze 87D4-87FF werden durch den Monitor belegt. Der Stack beginnt bei 87D3.
Nachfolgend nun eine Aufstellung der möglichen Befehle:
Display (DP)
Format: nnnnDP
(nnnn = 4stellige Adresse)
Ausgabe: Inhalt der Speicherstelle nnnn wird auf Ziffer 5 und 6 angezeigt.

```
oooo: 31 d3 87 3e 81 d3 23 dd 21 d6 87 dd 36 oo ef dd
oo1o: 36 o1 5f dd 36 o2 dc dd 36 o3 cd c3 23 oo oo oo
oo2o: c3 5d o3 dd 36 o4 49 dd 36 o5 d9 dd 36 o6 e7 af
oo3o: 32 f1 87 32 df 87 32 eo 87 32 dd 87 3e o3 32 de
oo4o: 87 oo oo oo oo oo oo cd ec oo cd 83 o3 3e oo d3
oo5o: 22 cd 97 oo 7b fe o2 ca cf o1 fe o3 ca d7 o1 fe
oo6o: o4 ca 7b o2 fe o5 ca e3 o1 fe o6 ca 7e o2 fe o7
oo7o: ca be o2 fe o8 ca 4a o3 fe o9 ca 43 o3 fe oa ca
oo8o: fe o3 fe ob ca oo o5 c2 91 oo 21 eo 87 34 c3 4d
oo9o: oo cd d3 oo c3 8a oo cd bd oo db 22 b7 e2 a8 oo
ooao: 21 eo 87 af 77 c3 97 oo 21 eo 87 7e b7 c2 97 oo
oobo: cd bd oo db 22 b7 ea 97 oo cd oa o1 c9 af 47 57
ooco: 5f 21 e1 87 13 cd fa oo 23 eb 29 eb 7b fe 4o c2
oodo: c5 oo c9 o6 oo 21 de 87 4e 21 e1 87 o9 73 od 79
ooeo: fe ff c2 e7 oo 3e o3 21 de 87 77 c9 o6 o6 21 e1
oofo: 87 3e oo 77 o5 c8 23 c3 f1 oo 7b d3 2o 7e d3 21
o1oo: oe 44 od 2o fd oo oo oo oo c9 c5 e5 21 9f o1 3a
o11o: df 87 b7 ca 1b o1 o6 oo oe 1o o9 3e oo 32 df 87
o12o: o6 oa 3e eo d3 22 db 22 o5 c2 22 o1 e6 of fe of
o13o: c4 7c o1 23 23 23 23 o6 oa 3e do d3 22 db 22 o5
o14o: c2 39 o1 e6 of fe of c4 7c o1 23 23 23 23 o6 oa
o15o: 3e bo d3 22 db 22 o5 c2 5o o1 e6 of fe of c4 7c
o16o: o1 23 23 23 23 o6 oa 3e 7o d3 22 db 22 o5 c2 67
o17o: o1 e6 of fe of c4 7c o1 7b e1 c1 c9 fe oe c2 85
o18o: o1 7e c3 9d o1 23 fe od c2 8f o1 7e c3 9d o1 23
o19o: fe ob c2 99 o1 7e c3 9d o1 23 fe o7 7e 5f c9 o2
o1ao: o3 o4 o5 o6 o7 o8 o9 cb 7b 7f a8 bf 88 f6 fa o2
o1bo: oa ob oo oo oo oo oo 35 dc 77 67 ff fb ef 5f bf
o1co: 88 f6 fa cb 7b 7f a8 ff fb ef 5f 35 dc 77 67 3e
o1do: o2 32 df 87 c3 8a oo cd ec oo 21 de 87 3e o3 77
o1eo: c3 8a oo cd ef o1 cd oc o2 cd 27 o2 c3 8a oo 16
o1fo: o6 fd 21 e7 87 dd 21 e1 87 21 bf o1 cd 52 o2 78
o2oo: fd 77 oo dd 23 fd 23 15 c2 f9 o1 c9 fd 21 e7 87
o21o: fd 7e oo 47 fd 7e o1 cd 6o o2 6f fd 7e o2 47 fd
o22o: 7e o3 cd 6o o2 67 c9 7e 47 cd 6a o2 fd 21 bf o1
o23o: dd 21 e1 87 o6 oo 4d fd o9 fd 7e oo dd 77 o4 fd
o24o: 21 bf o1 4c fd o9 fd 7e oo dd 77 o5 3e o1 32 dd
o25o: 87 c9 oo o6 oo dd 7e oo 4e b9 c8 o4 23 c3 58 o2
o26o: cb 27 cb 27 cb 27 cb 27 8o c9 47 cb 2f cb 2f cb
o27o: 2f cb 2f e6 of 67 78 e6 of 6f c9 c3 oo oo 3a dd
o28o: 87 fe o1 c2 8a oo cd 8c o2 c3 e3 o1 cd oc o2 23
o29o: c5 7d 54 47 cd 29 o2 dd 21 e1 87 dd 7e o4 dd 77
o2ao: oo dd 7e o5 dd 77 o1 7a 47 cd 29 o2 dd 21 e1 87
o2bo: dd 7e o4 dd 77 o2 dd 7e o5 dd 77 o3 c1 c9 3e oo
o2co: d3 22 21 eo 87 34 cd 97 oo 7b fe o2 c2 d7 o2 3e
o2do: o2 32 df 87 c3 be o2 fe o6 ca 19 o3 fe o4 ca 7b
o2eo: o2 oe o5 21 de 87 71 cd d3 oo 3e oo d3 22 21 eo
o2fo: 87 34 cd 97 oo oe o4 21 de 87 71 7b fe o2 c2 o9
o3oo: o3 3e o2 32 df 87 c3 ea o2 fe o4 ca 7b o2 fe o6
o31o: ca 19 o3 cd d3 oo c3 be o2 cd ef o1 cd oc o2 54
o32o: fd 21 e7 87 fd 23 fd 23 cd 1b o2 7c 62 77 3e o3
o33o: 11 de 87 12 cd 8c o2 cd ef o1 cd oc o2 cd 27 o2
o34o: c3 be o2 cd ef o1 cd oc o2 e9 cd ef o1 cd oc o2
o35o: 7e 32 ed 87 22 ee 87 3e e7 77 c3 8a oo 33 33 ed
o36o: 73 fc 87 3b 3b e5 f5 3a ed 87 2a ee 87 77 f1 e1
o37o: cd 94 o3 cd ec oo af 32 ee 87 32 ef 87 32 ed 87
o38o: c3 oo oo dd 21 e1 87 dd 36 oo 4c dd 36 o1 1c dd
o39o: 36 o2 44 c9 ed 43 f2 87 ed 53 f4 87 22 f6 87 dd
o3ao: 22 f8 87 fd 22 fa 87 32 fo 87 16 o7 dd 21 fo 87
o3bo: o1 d6 87 dd 6e oo dd 66 o1 dd e5 d5 cd 9o o2 af
o3co: 32 e5 87 d3 22 oa 32 e6 87 c5 cd 97 oo 7b fe o6
o3do: c2 ca o3 21 eo 87 34 c1 d1 dd e1 o3 dd 23 dd 23
o3eo: 15 c2 b3 o3 cd ec oo 3a fo 87 ed 4b f2 87 ed 5b
o3fo: f4 87 2a f6 87 dd 2a f8 87 fd 2a fa 87 c9 3e cf
o4oo: d3 o3 3e 7f d3 o3 cd 5o o4 7e 4f cd 1a o4 23 1b
o41o: 7b b2 2o f5 c3 oo oo oo oo oo f5 c5 e5 79 f5 cd
o42o: 27 o4 f1 e1 c1 f1 c9 37 o6 o9 f5 d4 3c o4 dc 41
o43o: o4 f1 1f 1o f5 cd 3c o4 cd 3c o4 c9 af d3 o1 18
o44o: o6 3e ff d3 o1 18 oo 21 16 oo 2b 7c b5 2o fb c9
o45o: 21 e1 87 af 77 23 77 23 77 23 77 23 3e af 77 23
o46o: 3e ef 77 cd bd oo af d3 22 cd 97 oo 7b fe o2 ca
o47o: 9o o4 fe o9 ca 7f o4 fe o6 ca 98 o4 c2 86 o4 21
o48o: eo 87 34 c3 66 o4 cd d3 oo 21 eo 87 34 c3 66 o4
o49o: 3e o2 32 df 87 c3 7f o4 cd ef o1 cd oc o2 eb d5
o4ao: 21 e1 87 af 77 23 77 23 77 23 77 23 3e af 77 23
o4bo: 3e 77 77 cd bd oo oo oo oo oo af d3 22 cd 97 oo
o4co: 7b fe o2 ca e7 o4 fe o6 ca d6 o4 fe o9 ca fo o4
o4do: oo oo oo c2 dd o4 21 eo 87 34 c3 ba o4 cd d3 oo
o4eo: 21 eo 87 34 c3 ba o4 3e o2 32 df 87 c3 d6 o4 oo
o4fo: cd ef o1 cd oc o2 d1 d5 ed 52 d1 eb c9 oo oo oo
o5oo: 3e cf d3 o3 3e ff d3 o3 oo cd 5o o4 cd 2o o5 77
o51o: 23 1b 7b b2 ca oo oo o1 of oo cd 44 o5 c3 oc o5
o52o: d9 db o1 17 3o fb o6 o8 11 ob oo cd 3e o5 11 15
o53o: oo cd 3e o5 db o1 17 cb 19 1o f3 79 d9 c9 1b 7b
o54o: b2 2o fb c9 ob 79 bo 2o fb c9 oo oo oo oo oo oo
```

Bild 4. Das Monitorprogramm ist nur knapp 1,5 KByte lang

Enter (EN)
Format: EN
Ausgabe: Inhalt der Speicherstelle nnnn+1 auf Ziffer 5 und 6. Speicherstelle nnnn+1 auf Ziffern 0...3. Abbruch durch Drücken der Break-Taste.

Insert (I)
Format: nnnnImmEN
(mm = Hexadezimalzahl)
Ausgabe: mm wird in die nnnn Speicherzelle übernommen. Anzeige der Speicherstelle nnnn+1 auf den Ziffern 0...3. Inhalt der Speicherstelle nnnn+1 auf Ziffer 5 und 6. Wenn keine Break-Taste gedrückt ist, erwartet der EMUF die Eingabe der nächsten Hexadezimalzahl für die angezeigte Speicherstelle.

Clear (CR)
Format: CR
Ausgabe: Display wird gelöscht.

Break (BK)
Format: BK
Ausgabe: EMUF meldet sich mit 'r u n'.

Go (G)
Format: nnnnG
Ausgabe: Programm wird bei Adresse nnnn gestartet.

Breakpoint (BP)
Format: nnnnBP
Ausgabe: Bei Adresse nnnn wird ein Breakpoint gesetzt (RST20). Beim Erreichen des Breakpoint wird eine Routine angesprungen, die die Register A, BC, DE, HL, IX, IY und SP anzeigt. Dabei erscheint auf Ziffer 6 eine Abkürzung für das entsprechende Register. Durch Drücken der Enter-Taste wird das nächste Register angezeigt. Nach dem letzten

Register wird der Breakpoint wieder gelöscht, und der Monitor meldet sich mit 'r u n'.

In (IN)
Serielle Eingabe von Daten über Bit 7 des PIO 0 Port B

Format: IN
Ausgabe: Auf Ziffer 4 und 5 erscheint AN. Monitor erwartet Anfangsadresse für die Ablage der aufzunehmenden Daten.

Format: nnnnEN
Ausgabe: Auf Ziffer 4 und 5 erscheint EN. Monitor erwartet Endadresse für die Ablage der aufzunehmenden Daten.
Format: nnnnG

Bis alle Daten eingelesen sind, erscheint auf Ziffer 5 ein E.
Voraussetzung für eine ordnungsgemäße Eingabe ist ein entsprechendes Programm beim die Daten sendenden Computer. Die Übertragungsrate beträgt etwa 2400 Baud.

Out (O)
Serielle Ausgabe von Daten über Bit 7 des PIO 0, Port B

Format: 0
Ausgabe: Analog zur Eingabe von Daten.

Erweiterungen sind möglich

Der Monitor belegt ungefähr 1,5 KByte des 2-KByte-EPROMs (2716). Es ist durchaus denkbar, die fünf noch freien Tasten mit weiteren Funktionen zu belegen. Aber auch auf der vorgestellten Stufe ermöglicht der Monitor zusammen mit der Tastatur und dem Display einen recht flexiblen Einsatz des Z80-EMUF.

Literatur

[1] *Kanis, Wolfgang:* Der Z80-EMUF. mc 1983, Heft 4, S. 112.

Joachim König

Z80-EMUF als universelle Fernbedienung

Fertig zu kaufende Fernsteuerungen sind zwar heute schon recht preiswert zu haben, lassen sich aber kaum an individuelle Bedürfnisse anpassen. Wir zeigen Ihnen, wie man mit modernen Bausteinen eine Fernsteuerung aufbaut, die – dank Z80-EMUF – auf Tastendruck auch komplexe Abläufe bewältigt.

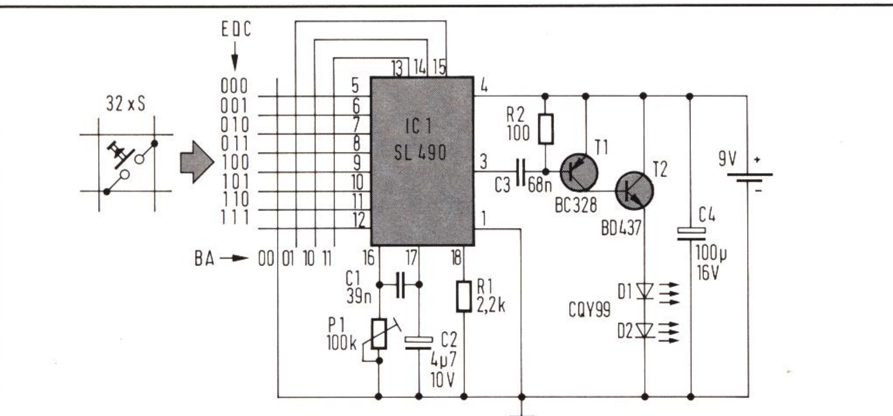


Bild 1. Der Sender wird mit einer 9-V-Batterie versorgt – ein Schalter ist nicht erforderlich

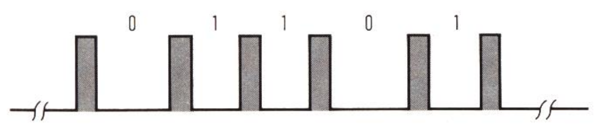


Bild 2. Das Datenwort besteht aus sechs kurzen Impulsen, deren Zwischenräume die Information darstellen. Ein kurzer Zwischenraum bedeutet „1“, ein langer „0“. Hier ist das Signal für das Datenwort 01101 (hex 0D) dargestellt

Das Kernstück des Senders ist die integrierte Schaltung SL 490 (*Bild 1*). In ihr ist die ganze Logik enthalten, die zum Betrieb des Senders nötig ist. Sie decodiert das Tastenfeld und erzeugt ein PPM-Signal (Puls-Pausen-Modulation).

Es können maximal 32 Tasten angeschlossen werden. Jede Taste entspricht einem fünfstelligen Binärcode. Drückt man eine Taste, so erscheinen am Ausgang des SL 490 sechs kurze Impulse.

Die Pause zwischen diesen Impulsen enthält die Information. Eine kurze Pause bedeutet, „logisch 1“, eine lange „logisch 0“ (*Bild 2*). Die beiden als Darlington geschalteten Transistoren T1 und T2 dienen als Verstärker und steuern direkt die Infrarotsender D1 und D2. Der SL 490 besitzt eine Abschaltautomatik, so daß nur wenige µA Strom fließen, wenn keine Taste gedrückt ist. Deswegen ist kein Schalter in der Stromversorgung nötig. Als Stromversorgung genügt eine einfache 9-V-Batterie. Mit P1 kann man die Zeit zwischen den Impulsen justieren. Das Verhältnis zwischen „1“ und „0“ beträgt 1:1,5.

Der Empfänger

Das mit der Empfängerdiode D1 gewonnene Signal wird mit dem Operationsverstärker SL 480 verstärkt (*Bild 3*).

Dann wird es über T2 dem Schmitt-Trigger IC2 zugeleitet. Der Schmitt-Trigger macht saubere Rechteckimpulse aus dem Signal, und das Monoflop IC3 verlängert diese Impulse auf 100 µs. Da das Monoflop nicht nachgetriggert werden kann, werden Störimpulse ausgeblendet. Am Ausgang von IC3 steht eine Impulsfolge, wie in Bild 2 gezeigt, bereit. Diese Impulse werden an Port A, Bit 0 geführt. Mit dem Potentiometer P1 kann die Empfindlichkeit eingestellt werden.

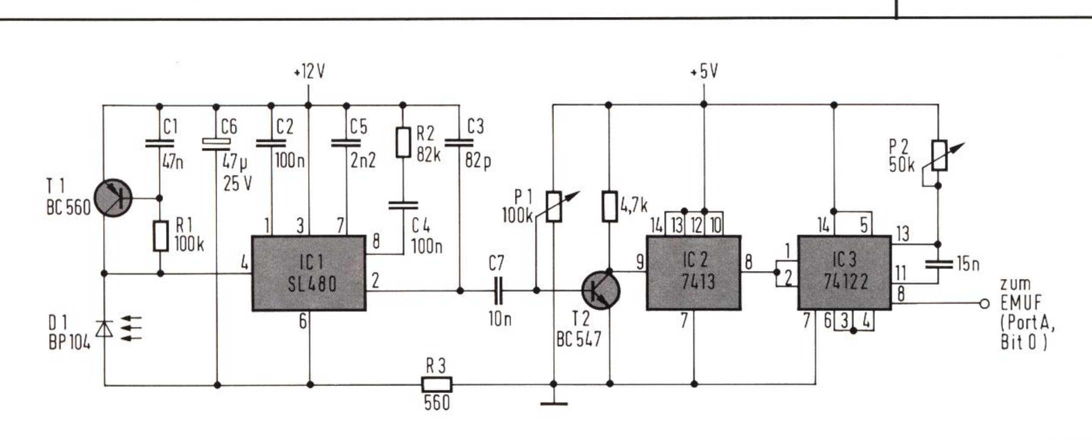


Bild 3. Der Empfänger liefert am Ausgang saubere Signale gemäß Bild 2. Störimpulse unterdrückt er

```
                         .z80
0002             portlc  equ    02h             ; port 1 control
0000             portld  equ    00h             ; port 1 data
0000'  31 87FF   start:  ld     sp,087ffh
0003'  21 0147'          ld     hl,inittab      ; ports initialisieren
0006'  06 04             ld     b,4             ; anzahl der ports
0008'  4E        st1:    ld     c,(hl)          ; hole portadresse (control)
0009'  23                inc    hl
000A'  7E                ld     a,(hl)
000B'  ED 79             out    (c),a           ; setze mode
000D'  23                inc    hl
000E'  7E                ld     a,(hl)
000F'  ED 79             out    (c),a           ; bestimme input und output bits
0011'  3E 03             ld     a,3
0013'  ED 79             out    (c),a           ; disable interupts
0015'  0D                dec    c
0016'  0D                dec    c               ; mache datenadressse
0017'  3E 00             ld     a,0
0019'  ED 79             out    (c),a           ; setze ausgaenge auf 0
001B'  23                inc    hl
001C'  10 EA             djnz   st1
001E'  3E 00             ld     a,0
0020'  32 0004"          ld     (port1m),a      ; reset memoryimage
0023'  32 0005"          ld     (port2m),a      ; reset memoryimage
0026'  32 0006"          ld     (port3m),a      ; reset memoryimage
0029'  32 0007"          ld     (port4m),a      ; reset memoryimage
002C'  21 0193'          ld     hl,lampe
002F'  22 0000"          ld     (table),hl      ; kontext = lampe
0032'  CD 00A9'  main1:  call   getword         ; hole nummer
0035'  1E 00     main:   ld     e,0             ; clear zaehler
0037'  50                ld     d,b             ; save nummer
0038'  1C                inc    e               ; erhoehe zaehler
0039'  CD 00A9'  m1:     call   getword         ; hole naechste nummer
003C'  7A                ld     a,d
003D'  B8                cp     b               ; neue nummer gleich der ersten
003E'  C2 0035'          jp     nz,main         ; wenn ungleich dann beginne neu
0041'  1C                inc    e               ; erhoehe zaehler
0042'  7B                ld     a,e
0043'  FE 05             cp     5               ; 5 gleiche nummern eingetroffen?
0045'  C2 0039'          jp     nz,m1           ; wenn nicht hole neachste nummer
0048'  CD 004E'          call   schalt          ; fuehre funktion aus
004B'  C3 0032'          jp     main1
004E'  CD 007A'  schalt: call   convert         ; wandle physische nummer in logische um
0051'  7B                ld     a,e
0052'  FE 10             cp     16              ; if nummer>16 goto switch
0054'  D2 0064'          jp     nc,switch
0057'  2A 0000"          ld     hl,(table)      ; get basis fuer verzweigung
005A'  CB 23             sla    e               ; e = e * 2
005C'  16 00             ld     d,0
005E'  19                add    hl,de           ; hl = basis + index
005F'  5E                ld     e,(hl)          ; hole verzweigungadresse
0060'  23                inc    hl
0061'  56                ld     d,(hl)
0062'  EB                ex     de,hl
0063'  E9                jp     (hl)            ; springe zur routine
0064'  21 0173'  switch: ld     hl,basis        ; basisadressse der kontexttabellenadress
0067'  7B                ld     a,e             ; a = tastennummer
0068'  E6 0F             and    0fh             ; a = tastennummer-16
006A'  5F                ld     e,a
006B'  CB 23             sla    e
006D'  16 00             ld     d,0             ; de = tastennummer*2
006F'  19                add    hl,de           ; hl = basis+tastennummer*2
```

```
0070'  5E                ld     e,(hl)          ; get low byte
0071'  23                inc    hl
0072'  56                ld     d,(hl)          ; get high byte
0073'  ED 53 0000"       ld     (table),de      ; set new kontextbasis
0077'  C3 0083'          jp     endret
007A'  21 0153'  convert:ld     hl,contab       ; adresse der converttabelle
007D'  16 00             ld     d,0
007F'  58                ld     e,b             ; pyhsische tastennummer nach e
0080'  19                add    hl,de           ; hl = contab + physische tastennummer
0081'  5E                ld     e,(hl)          ; log tastennummer nach e
0082'  C9                ret
0083'  21 01F4   endret: ld     hl,0500         ; startwert fuer time-out counter
0086'  50                ld     d,b             ; save taste
0087'  CD 0091'  er1:    call   getwo1          ; hole naechste taste
008A'  C8                ret    z               ; time-out
008B'  7A                ld     a,d
008C'  B8                cp     b               ; gleiche taste?
008D'  C0                ret    nz              ; andere taste gibt auch time-out
008E'  C3 0083'          jp     endret
0091'  E5        getwo1: push   hl
0092'  21 0091'          ld     hl,getwo1
0095'  22 0002"          ld     (retadr),hl     ; setze return adresse fuer fehlerfall
0098'  E1                pop    hl
0099'  06 00             ld     b,0             ; clear nummer
009B'  DB 00     g10:    in     a,(portld)      ; warte auf eine 1
009D'  E6 01             and    1
009F'  C2 00B8'          jp     nz,getw2
00A2'  2B                dec    hl
00A3'  7C                ld     a,h
00A4'  B5                or     l
00A5'  C8                ret    z               ; time-out
00A6'  C3 009B'          jp     g10
00A9'  21 00A9'  getword:ld     hl,getword
00AC'  22 0002"          ld     (retadr),hl     ; setze returnadresse fuer fehlerfall
00AF'  06 00             ld     b,0             ; clear nummer
00B1'  DB 00     getw1:  in     a,(portld)      ; warte auf 1
00B3'  E6 01             and    1               ; check ob bit = 1
00B5'  CA 00B1'          jp     z,getw1
00B8'  CD 00D2'  getw2:  call getbit            ; hole 1.bit
00BB'  CB 10             rl b                   ; schiebe bit nach b
00BD'  CD 00D2'          call getbit
00C0'  CB 10             rl b                   ; 2.bit
00C2'  CD 00D2'          call getbit
00C5'  CB 10             rl b                   ; 3.bit
00C7'  CD 00D2'          call getbit
00CA'  CB 10             rl b                   ; 4.bit
00CC'  CD 00D2'          call getbit
00CF'  CB 10             rl b                   ; 5.bit
00D1'  C9                ret
00D2'  E5        getbit: push   hl
00D3'  C5                push   bc
00D4'  01 01F4           ld     bc,500          ; init timeout counter
00D7'  21 0000           ld     hl,0            ; clear counter
00DA'  DB 00     g1:     in     a,(portld)      ; warte auf low flanke
00DC'  E6 01             and    1
00DE'  C2 00DA'          jp     nz,g1
00E1'  DB 00     getb3:  in     a,(portld)      ; warte auf high flanke
00E3'  E6 01             and    1
00E5'  C2 00F2'          jp     nz,getb2
00E8'  23                inc    hl              ; erhoehe zaehler
00E9'  0B                dec    bc              ; decrement time out counter
00EA'  78                ld     a,b             ; wenn timeout counter = 0
00EB'  B1                or     c               ; dann springe auf fehlerroutine
```

Bild 4. Das Auswerteprogramm: Wer keinen Linker besitzt, muß die mit doppelten Anführungszeichen gekennzeichneten Adressen von Hand verändern

```
00EC'  CA 011A'            jp     z,getb7
00EF'  C3 00E1'            jp     getb3
00F2'  7D           getb2: LD     A,l          ; if l in [130..200]
00F3'  FE C9               CP     200+1        ; dann ist es eine 0
00F5'  D2 0100'            JP     NC,.L1
00F8'  FE 82               CP     130
00FA'  DA 0100'            JP     C,.L1
00FD'  C3 0115'     .L2:   jp  getb4
0100'  7D           .L1:   ld     a,l          ; if l in [080..130]
0101'  FE 83               CP     130+1        ; dann ist es eine 1
0103'  D2 010E'            JP     NC,.L3
0106'  FE 50               CP     080
0108'  DA 010E'            JP     C,.L3
010B'  C3 0111'     .L4:   jp  getb6
010E'  C3 011A'     .L3:   jp getb7            ; error in bitlaenge
0111'  37           getb6: scf                 ; es war eine 1
0112'  C1                  pop    bc
0113'  E1                  pop    hl
0114'  C9                  ret
0115'  37           getb4: scf
0116'  3F                  ccf                 ; es war eine 0
0117'  C1                  pop    bc
0118'  E1                  pop    hl
0119'  C9                  ret
011A'  C1           getb7: pop    bc
011B'  E1                  pop    hl
011C'  FD E1               pop    iy           ; dummy pop der ret addresse
011E'  2A 0002"            ld     hl,(retadr)
0121'  E9                  jp     (hl)         ; error return
                    ; Routinen fuer lampe
                    ; Beispiel fuer die Realisation einer TASTE
0122'  3A 0004"     lampe1: ld    a,(port1m)   ; get memoryimage
0125'  CB CF               set    1,a          ; set bit-1
0127'  D3 00               out    (port1d),a   ;
0129'  32 0004"            ld     (port1m),a
012C'  CD 0083'            call   endret       ; wait auf ende tastendruck
012F'  3A 0004"            ld     a,(port1m)
0132'  CB 8F               res    1,a          ; reset bit-1
0134'  D3 00               out    (port1d),a   ;
0136'  32 0004"            ld     (port1m),a
0139'  C9                  ret
                      ; Beispiel fuer die Realisatin eines SCHALTERS
013A'  3A 0004"     lampe2: ld    a,(port1m)   ; get memoryimage
013D'  EE 04               xor    4h           ; toggle bit-2
013F'  D3 00               out    (port1d),a
0141'  32 0004"            ld     (port1m),a   ; set memoryimage
0144'  C3 0083'            jp     endret       ; warte auf ende tastendruck und ret nach main
                    ; daten im eprom
0147'  03 FF 01 13  inittab:db    3,0ffh,01,13h,0ffh,0,2,0ffh,0,12h,0ffh,0
014B'  FF 00 02 FF
014F'  00 12 FF 00
                    ; umrechnungtabelle pyhsische taste im log taste
0153'  00 01 02 03  contab: db    0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21, 22,23
0157'  04 05 06 07
015B'  08 09 0A 0B
015F'  0C 0D 0E 0F
0163'  10 11 12 13
0167'  14 15 16 17
016B'  18 19 1A 1B          db    24,25,26,27,28,29,30,31
016F'  1C 1D 1E 1F
0173'  0193' 0193'  basis:  dw    lampe,lampe,lampe,lampe,lampe,lampe,lampe,lampe
0177'  0193' 0193'
017B'  0193' 0193'
017F'  0193' 0193'
0183'  0193' 0193'          dw    lampe,lampe,lampe,lampe,lampe,lampe,lampe,lampe
0187'  0193' 0193'
018B'  0193' 0193'
018F'  0193' 0193'
                    ; kontextadressen fuer lampen
0193'  0122' 013A'  lampe:  dw    lampe1,lampe2,endret,endret,endret,endret,endret,endret
0197'  0083' 0083'
019B'  0083' 0083'
019F'  0083' 0083'
01A3'  0083' 0083'          dw    endret,endret,endret,endret,endret,endret,endret,endret
01A7'  0083' 0083'
01AB'  0083' 0083'
01AF'  0083' 0083'
01B3'                       dseg
                    ; daten im ram
0000"               table:  ds    2
0002"               retadr: ds    2
0004"               port1m: ds    1            ; memory abbildung der ports
0005"               port2m: ds    1
0006"               port3m: ds    1
0007"               port4m: ds    1
                                  end
```

Mit P2 wird die Impulsbreite auf 100 µs eingestellt. Wer kein Oszilloskop besitzt, stellt P2 in Mittenstellung. P1 muß so eingestellt werden, daß in Ruhestellung (keine Impulse vom Sender) Low-Pegel am Eingang von IC2 anliegt.

Das Programm

Bei der Entwicklung des Programmes (*Bild 4*) wurde auf eine möglichst einfache Erweiterungsmöglichkeit der Funktion Wert gelegt. Die Software stellt nur die Auswertung der empfangenen Impulsfolgen und die davon abhängigen Verzweigungen dar. Die Routinen, die nötig sind, um die einzelnen Funktionen auszuführen, sind abhängig von der jeweiligen Funktion. Als Beispiel sind jedoch zwei Routinen abgedruckt, die eine Taste bzw. einen Schalter emulieren. Der Programmteil ENDRET wartet auf das Loslassen der gedrückten Taste, so daß im Programm eine Funktion nicht mehrmals ausgeführt wird, wenn man die Taste festhält.

Von den 32 zur Verfügung stehenden Tastencodes werden 16 benutzt, um die Aufgabe der Fernbedienung umzuschalten. So kann man 16 Aufgaben mit jeweils 16 Funktionen, also 256 Funktionen, ausführen. In der Tabelle BASIS stehen die 16 Adressen der jeweiligen Funktionsbasisadressen.

Ein Beispiel:
Wenn in der Variablen TABLE die Adresse der Tabelle LAMPE steht, werden beim Druck auf eine der Tasten 0...15 die jeweiligen Adressen aus der Tabelle LAMPE geholt, und das Programm wird dort ausgeführt. Eine Erweiterung der Funktionen ist also dadurch möglich, daß man die Adresse der neuen Funktion in die entsprechende Tabelle einbaut, ohne das Kernprogramm zu ändern.

Die Routine CONVERT dient zum Umwandeln des physikalischen Tastencodes in den logischen. Je nach Verdrahtung des Tastenfeldes ergeben sich die übertragenen Tastencodes. Um nun die Codes in eine einheitliche Reihenfolge zu bringen, müssen sie umgewandelt werden. Das abgedruckte Programm nimmt eine 1:1-Umwandlung vor.

Wem die 256 Funktionen zu wenig sind, der kann das Programm auch so umschreiben, daß eine der 32 Tasten die Umschaltung einleitet und die nächste Taste die Aufgabe festlegt. Dann sind insgesamt 992 Funktionen möglich.

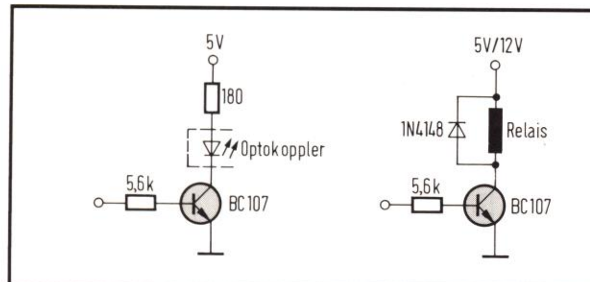


Bild 5. So lassen sich angesteuerte Geräte oder Schaltungen galvanisch vom EMUF-Ausgang trennen

In *Bild 5* sind einige Möglichkeiten der Steuerung gezeigt. Man kann mit den PIOs über einen Transistor direkt ein Relais ansteuern und damit 220-V-Geräte bedienen. Wichtig: Das gesteuerte Gerät und der EMUF müssen galvanisch getrennt sein. Falls die Reichweite der Fernbedienung (ca. 6 m) unbefriedigend ist, kann man die Sendedioden in Reflektoren setzen.

Linken des Programmes

Das Listing ist so geschrieben, daß die Adressen des Programmes erst beim Linken festgelegt werden. Dazu ist ein Linker nötig, der das Programm nicht ablauffähig in den Speicher legt. Der Linker von Pascal MT+ ist dazu in der Lage. Um das Objektfile damit zu linken, bekommt es den Namen FERNB.ERL; dann lautet der Aufruf LINK FERNB/P:0/D:8000. Damit wird das Programm auf Adresse 0 gelegt, die Daten auf Adresse 8000h. Wer keinen Linker besitzt, muß alle Adressen, die mit einem doppelten Anführungszeichen gekennzeichnet sind, entsprechend verändern.

Literatur

[1] Remote Control Data, Plessey Semiconductors.
[2] IR-Fernsteuerung. Elektor, Heft 139, Seite 97.

Erich Gaulke

Z80-EMUF als Spooler

Manche Ausgabegeräte, wie z. B. Typenraddrucker, sind recht langsam und halten den Computer unnötig auf. Ein Spooler (simultaneous peripheral output on-line) kann den auszugebenden Text schnell vom Computer aufnehmen, zwischenspeichern und mit der für den Drucker erforderlichen geringeren Geschwindigkeit weitergeben. Währenddessen kann sich der Computer schon wieder anderen Aufgaben zuwenden.

Ein Spooler ist, wie bereits in [1] beschrieben, im wesentlichen ein Speicher nach dem FIFO-Prinzip (first in, first out). Ein übliches RAM muß also im Sinne eines FIFOs verwaltet werden. Mit Hilfe des Z80-EMUF läßt sich das einfach und zugleich effizient erledigen und wurde hier für eine Centronics-Schnittstelle realisiert. Wenn man den Spooler im Gegensatz zu [1] als eigenständiges Mikrocomputersystem aufbaut, hat das drei Vorteile:

1. Unabhängig von der Druckertreiberroutine bzw. vom Anwenderprogramm (Textverarbeitung, Basic, Assembler, DOS usw.) bedient der Spooler den Drucker.
2. Der Speicherplatz des Hauptrechners wird nicht durch den Spooler belastet.
3. Durch Interruptsteuerung des Programms ist der Z80-EMUF mit der Druckersteuerung kaum ausgelastet und könnte daher auch noch viele andere Aufgaben als Co-Prozessor übernehmen.

Beschaltung des Z80-EMUF

Die PIOs [2] des Z80-EMUF [3] lassen sich im Input- bzw. Output-Mode betreiben. Dann stehen neben den acht Ein- bzw. Ausgabeleitungen zwei für die Abwicklung des Handshakes zur Verfügung. Damit entfällt die Abwicklung der Datenübergabe vom Prozessor aus; er wird für andere Aufgaben freigestellt. Allerdings sind die Handshake-Leitungen den Anforderungen der jeweiligen Druckerschnittstelle anzupassen. Im vorliegendem Fall galt es, den Druckerport eines TRS-80 und eine Typenradschreibmaschine Olympia ES 100 anzuschließen. *Bild 1* zeigt die Beschaltung: Außer den Invertern zur Anpassung der Busy- und Ready-Signale wird ein Monoflop benötigt, um die Daten auszugeben. Als Gatter-Logik kommen vorzugsweise CMOS-ICs in Betracht, da Gatterlaufzeiten hier nicht kritisch sind.

Das Spoolerprogramm

Bei der Entwicklung des Programms wurde auf einen möglichst effizienten Umgang mit Rechenzeit und Speicherplatzbedarf geachtet. Für die Realisierung des FIFOs (2036 Byte Kapazität) wird mit nur zwei 16-Bit-Zeigern in DE und HL gearbeitet; die weiteren Informationen über den Zustand des Systems sind platzsparend und mit kleiner Zugriffszeit als Flags im C-Register untergebracht. Durch die Nutzung der Vektor-Interrupts konnte vollständig auf die Verwendung von Warteschleifen verzichtet werden. Damit benötigt das System für die Bearbeitung der Ein- oder

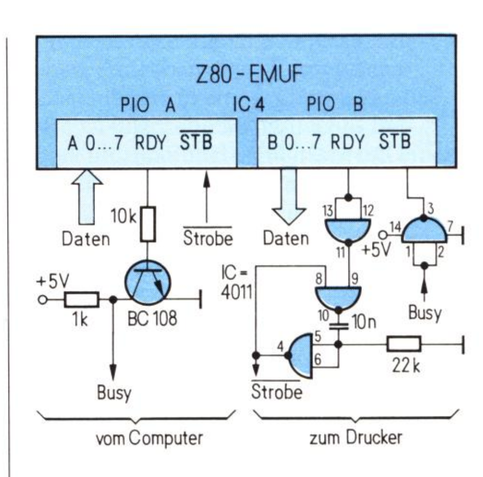


Bild 1. Beschaltung der Z80-PIO zur Verbindung mit Computer und Drucker am Beispiel des TRS-80

Ausgabe eines Zeichens unabhängig vom Tempo der angeschlossenen Geräte nur noch 100 µs bei einer Taktfrequenz von 2,5 MHz. Das führt bei Verwendung eines schnellen Matrixdruckers mit 100 Zeichen pro Sekunde zu einer Auslastung des Z80-EMUF von nur 2 %! Er könnte also noch viele weitere Aufgaben z. B. eines Kommunikationsrechners übernehmen.
Das vollständige Assembler-Listing ist in *Bild 2* abgedruckt: *Tabelle 1* zeigt die Steuerflags in übersichtlicher Darstellung. Nach dem Power-On-Reset beginnt der Prozessor bei Speicherstelle 0000 in dem ROM. Zunächst werden sämtliche PORTs mit den Daten aus der Tabelle TAB1 und unter Nutzung des Blockausgabebefehls OTIR initialisiert. Dann werden die Anfangswerte der Zeiger und Flags im C-Register geladen. Der nicht maskierbare Interrupt (NMI) am Prozessor gestattet es, den Druckvorgang zu unterbrechen und die Anfangswerte neu einzustellen.
Infolge der Interruptsteuerung besteht das Hauptprogramm HAUP aus einer leeren Schleife; hier ist noch Platz für Erweiterungen. Da das System nur durch Interrupts angestoßen wird, jedoch kein periodisches Interruptsignal

Tabelle: Die Steuerflags im C-Register

	Fall	EIN	AUS	Fertig	Zeichen da
Bit-Nr.	0	1	2	3	4
Bit = 1	a	Enable	Enable	ja	ja
Bit = 0	b	Disable	Disable	nein	nein
Anfangsstellung	1	1	0	1	0

```
0000          00030       ORG   0
0000 C30001   00040       JP    INIT
0008          00050       ORG   08H       ;ANFANG DER INTERRUPT-
0008 1F01     00060       DEFW  EIN       ;VEKTOREN
000A 6F01     00070       DEFW  AUS
0066          00080       ORG   66H       ;START NICHT MASKIERBARER
0066 C30001   00090       JP    INIT      ;INTERRUPT
              00100 ;
0010          00110 PI10  EQU   10H
8000          00120 ANF   EQU   8000H     ;RAM-ANFANG
87F3          00130 END   EQU   87FFH-12          ;PLATZ FUER STACK (=12)
              00140 ;
0100          00150       ORG   100H
0100 21C901   00160 INIT  LD    HL,TAB1   ;PORTS INITIALISIEREN
0103 4E       00170 INIA  LD    C,(HL)    ;PORT-ADRESSE LADEN
0104 23       00180       INC   HL
0105 46       00190       LD    B,(HL)    ;ZAHL DER BYTES LADEN
0106 23       00200       INC   HL
0107 EDB3     00210       OTIR            ;BYTES AUF PORT AUSGEBEN
0109 7E       00220       LD    A,(HL)
010A B7       00230       OR    A         ;TABELLENENDE WIRD DURCH NULL
010B 20F6     00240       JR    NZ,INIA   ;ERKANNT
010D 31FF87   00250       LD    SP,87FFH
0110 FB       00260       EI
0111 ED5E     00270       IM    2
0113 0E0B     00280       LD    C,11      ;INITIALISIERUNG DER FLAGS
0115 210080   00290       LD    HL,ANF    ;INITIALISIERUNG DER ZEIGER
0118 110080   00300       LD    DE,ANF
011B DB10     00310       IN    A,(PI10)    ;EING.-INT. ERMOEGLICHEN
011D 18FE     00320 HAUP  JR    HAUP      ;LEERES HAUPTPROGRAMM
              00330 ;
011F CB49     00350 EIN   BIT   1,C       ;EIN DISABLE ?
0121 280D     00360       JR    Z,EINE
0123 CD3501   00370       CALL  EIN1      ;ZEICHEN LESEN
0126 CB59     00380       BIT   3,C       ;FERTIG FLAG ?
0128 C48501   00390       CALL  NZ,AUS1   ;ZEICHEN AUSGEBEN WENN GESETZT
012B CB99     00400       RES   3,C       ;FERTIG FLAG ZURUECKSETZEN
012D FB       00410       EI              ;INTERRUPTS WIEDER ERMOEGLICHEN
012E ED4D     00420       RETI
0130 CBE1     00430 EINE  SET   4,C       ;ZEICHEN DA FLAG SETZEN
0132 FB       00440       EI
0133 ED4D     00450       RETI
0135 CD3C01   00460 EIN1  CALL  ABF1      ;ZEIGER U. FLAGS VERAENDERN
0138 CDBF01   00470       CALL  EIN3      ;EIN ZEICHEN LESEN
013B C9       00480       RET
013C CBD1     00490 ABF1  SET   2,C       ;ENABLE AUSGANG
013E 37       00500       SCF
013F CB41     00510       BIT   0,C       ;FALL A) ODER B)
0141 2822     00520       JR    Z,BFAL
0143 C5       00530       PUSH  BC
0144 E5       00540       PUSH  HL
0145 01F387   00550       LD    BC,END
0148 ED42     00560       SBC   HL,BC     ;HL-1=END ?
014A E1       00570       POP   HL
014B C1       00580       POP   BC
014C C0       00590       RET   NZ        ;ZURUECK WENN ENDE NICHT ERREICHT
014D CB81     00600       RES   0,C       ;FALL B) EIN
014F B7       00610       OR    A
0150 210080   00620       LD    HL,ANF
0153 ED52     00630       SBC   HL,DE     ;DE=ANF ?
0155 2002     00640       JR    NZ,BFA1
0157 CB89     00650       RES   1,C       ;DISABLE EIN, WENN DE=ANF
0159 21F487   00660 BFA1  LD    HL,END+1    ;LETZTES ZEICHEN IN END+1
015C CDBF01   00670       CALL  EIN3        ;ABLEGEN
015F 210080   00680       LD    HL,ANF      ;ZEIGER AUF ANFANG SETZEN
0162 DDE1     00690       POP   IX        ;ZURUECK ZUR INTERRUPT-ROUTINE
0164 C9       00700       RET
0165 EB       00710 BFAL  EX    DE,HL
0166 E5       00720       PUSH  HL
0167 ED52     00730       SBC   HL,DE     ;HL=DE-1 ?
0169 E1       00740       POP   HL
016A EB       00750       EX    DE,HL
016B C0       00760       RET   NZ        ;ZURUECK, WENN PLAETZE FREI
016C CB89     00770       RES   1,C       ;SONST DISABLE EIN
016E C9       00780       RET
              00790 ;
016F CB51     00800 AUS   BIT   2,C       ;AUS DISABLE ?
0171 280D     00810       JR    Z,AUSE
0173 CD8501   00820       CALL  AUS1      ;EIN ZEICHEN AUSGEBEN
0176 CB61     00830       BIT   4,C       ;ZEICHEN-DA-FLAG ?
0178 C43501   00840       CALL  NZ,EIN1   ;ZEICHEN LESEN, WENN GESETZT
017B CBA1     00850       RES   4,C       ;ZEICHEN-DA-FLAG ZURUECKSETZEN
017D FB       00860       EI
017E ED4D     00870       RETI
0180 CBD9     00880 AUSE  SET   3,C       ;FERTIG FLAG SETZEN
0182 FB       00890       EI
0183 ED4D     00900       RETI
0185 CD8C01   00910 AUS1  CALL  ABF2      ;FLAGS U. ZEIGER VERAENDERN
0188 CDC401   00920       CALL  AUS2      ;EIN ZEICHEN AUSGEBEN
018B C9       00930       RET
018C CBC9     00940 ABF2  SET   1,C       ;ENABLE EIN
018E 37       00950       SCF
018F CB41     00960       BIT   0,C       ;FALL A) ODER B)
0191 2808     00970       JR    Z,BFBL
0193 E5       00980       PUSH  HL
0194 ED52     00990       SBC   HL,DE     ;DE=HL-1 ?
0196 E1       01000       POP   HL
0197 C0       01010       RET   NZ        ;ZURUECK, WENN PLAETZE FREI
0198 CB91     01020       RES   2,C       ;DISABLE AUS
019A C9       01030       RET
019B D5       01040 BFBL  PUSH  DE        ;FALL B)
019C D9       01050       EXX
019D E1       01060       POP   HL
019E 01F387   01070       LD    BC,END
01A1 ED42     01080       SBC   HL,BC     ;DE-1=END ?
01A3 D9       01090       EXX
01A4 C0       01100       RET   NZ        ;ZURUECK, WENN ENDE NICHT ERREICHT
01A5 CBC1     01110       SET   0,C       ;FALL A) EIN
01A7 E5       01120       PUSH  HL
01A8 B7       01130       OR    A
01A9 110080   01140       LD    DE,ANF
01AC ED52     01150       SBC   HL,DE     ;HL=ANF ?
01AE 2002     01160       JR    NZ,BFB1
01B0 CB91     01170       RES   2,C       ;DISABLE AUS, WENN NICHTS FREI
01B2 11F487   01180 BFB1  LD    DE,END+1          ;ZEICHEN BEI END+1
01B5 CDC401   01190       CALL  AUS2              ;AUSGEBEN
01B8 110080   01200       LD    DE,ANF            ;ZEIGER AUF ANF SETZEN
01BB E1       01210       POP   HL
01BC DDE1     01220       POP   IX        ;ZURUECK ZUR INTERRUPT-ROUTINE
01BE C9       01230       RET
              01240 ;
01BF DB10     01250 EIN3  IN    A,(PI10)          ; PORT LESEN
01C1 77       01260       LD    (HL),A            ;IN BUFFER SPEICHERN
01C2 23       01270       INC   HL                ;ZEIGER WEITERZAEHLEN
01C3 C9       01280       RET
01C4 1A       01290 AUS2  LD    A,(DE)            ;ZEICHEN AUS BUFFER
01C5 D311     01300       OUT   (PI10+1),A        ;ZEICHEN AUSGEBEN
01C7 13       01310       INC   DE                ;ZAEHLER WEITER
01C8 C9       01320       RET
01C9 12       01330 TAB1  DEFB  PI10+2    ;INITIALISIERUNGSTABELLE
01CA 03       01340       DEFB  3         ;FUER DIE PORTS
01CB 08       01350       DEFB  8
01CC 4F       01360       DEFB  4FH
01CD 87       01370       DEFB  87H
01CE 13       01380       DEFB  PI10+3
01CF 03       01390       DEFB  3
01D0 0A       01400       DEFB  0AH
01D1 0F       01410       DEFB  0FH
01D2 87       01420       DEFB  87H
01D3 00       01430 ZEND  DEFB  0         ;ENDE WIRD DURCH NULL ERKANNT
0000          01440       END
```

Bild 2. Das vollständige Assembler-Listing mit Kommentaren

generiert wird, ist durch sorgfältige Programmierung dafür zu sorgen, daß es nicht stehenbleibt. Die Interrupt-Service-Routinen sind symmetrisch für Ein- und Ausgabe und werden von den jeweils zuständigen PIOs gestartet. Nachdem ein Zeichen in den PIO eingelesen ist, beginnt das Programm bei EIN. Wenn noch Plätze im Puffer frei sind, gilt EIN-enable. In EIN1 wird ein Zeichen eingelesen und abgespeichert, die Zeiger und Flags werden verwaltet. Das FERTIG-Flag ist nur gesetzt, wenn der Drucker sämtliche Zeichen im Buffer bereits ausgegeben hat und ein AUS-Interrupt kein weiteres Zeichen mehr vorfand. Der AUS-Kanal bleibt dann vollständig abgeschaltet, bis er nach Eingabe eines neuen Zeichens und Aufrufen von AUS1 in EIN vermittels des FERTIG-Flags reaktiviert wird.
Entsprechend wird bei vollem Puffer zunächst EIN-disable gesetzt und beim nächsten Interrupt auf der Eingangsseite, ohne daß inzwischen Platz frei wurde, das Zeichen-da-Flag gesetzt. Dadurch ist der Eingangskanal vollständig abgeschaltet, bis er nach Ausgabe eines Zeichens reaktiviert wird.

Erweiterungen möglich

Da das System nicht ausgelastet ist, bietet es sich an, Erweiterungen anzubringen. Dabei wäre z. B. denkbar, Umcodierungen der zu druckenden Zeichen vorzunehmen, Umlaute darzustellen, Matrix-Drucker mit Einzelpunktansteuerung zum Zeichnen von Kurven mit einem leistungsfähigen Befehlssatz zu versehen, Aufgaben eines Kommunikationsrechners in einem komplexen System zu übernehmen (einfaches Beispiel: bidirektionales Interface mit einer Schreibmaschine), oder an den zweiten PIO eine Tastatur mit Sonderfunktionen anzubringen.
(Herrn Hermann Wacker aus Braunschweig sei für Tips gedankt, die die schnelle Realisierung des Projekts ermöglichten.)

Literatur

[1] *Breymann, U.:* Druckerausgabe nebenbei. mc 1982, Heft 6. Seite 64
[2] Z80-PIO, Product Specification. Zilog
[3] *Kanis, W.:* Der Z80-EMUF. mc 1983, Heft 4, Seite 112

Dr. Sieghard Schicktanz

Multitasking mit dem Z80-EMUF

Mit einigen Einschränkungen im Komfort läßt sich ein Systemkern für Multitasking bereits mit einem Einplatinencomputer minimalen Ausbaus realisieren, wie ihn z. B. der Z80-EMUF darstellt. Damit kann ein vollständiges Anwendungsprogramm in einem EPROM-Bereich von etwa 2 KByte Platz finden. Der Beschreibung des Systemkerns ist eine kurze Besprechung der Grundlagen für ein Multitaskingsystem vorangestellt.

Multitasking, gelegentlich – nicht ganz korrekt – auch Mehrprogrammbetrieb genannt, wird häufig in der Prozeßsteuerungstechnik angewandt. Man versteht darunter die Bearbeitung einer komplexen Aufgabe durch mehrere, quasi parallel ablaufende Teilprogramme, die weitgehend unabhängig voneinander jeweils einen überschaubaren Teil der Gesamtaufgabe ausführen.
Durch die Aufteilung erreicht man eine wesentliche Vereinfachung der Programmerstellung. Eine geschickte Verteilung der Aufgaben ermöglicht es, die Teilprogramme sehr einfach aufzubauen und – wichtig bei industriellem Einsatz – unabhängig voneinander zu erstellen, da die Schnittstellen einfach gehalten werden können.
Außerdem können alle Koordinationsaufgaben in einem zentralen Teilprogramm (Scheduler) zusammengefaßt werden, das unabhängig vom jeweiligen Einsatzfall ausgeführt ist und darüber hinaus weitgehend die Unabhängigkeit der einzelnen anwendungsbezogenen Teilprogramme (Tasks) unterstützt. Der Systemkern stellt alle benötigten Hilfsroutinen zur Verfügung, die zum (quasi) parallelen Bearbeiten der Teilprogramme gebraucht werden, ohne daß der Programmersteller dies besonders berücksichtigen muß.
Bei großen Prozeßrechensystemen stellt diese Funktionen bereits das dort eingesetzte Betriebssystem zur Verfügung. Der Scheduler macht allerdings nur einen kleinen Teil des Betriebssystems aus; mit einigen Einschränkungen im Komfort kann ein Multitaskingsystem, samt Anwendungsprogramm, bereits auf einem Minimalsystem wie dem Z80-EMUF [1] in nur 2 KByte EPROM (und 2 KByte RAM) realisiert werden.

Was ist Multitasking?

Die Grundfunktionen eines Multitaskingsystems sind im folgenden kurz erläutert. Jede Task, d. i. jedes Teilprogramm, das eine abgeschlossene Teilaufgabe bearbeitet, kann verschiedene Zustände einnehmen:

1. Nicht vorhanden
2. Angehalten
3. Bereit (aktiv)
4. Wartend

Der Zustand 1, nicht vorhanden, ist hier trivial, da in einem EPROM-System Tasks nicht neu hinzukommen oder gelöscht werden können. In diesem Fall sind alle möglichen Tasks also immer im System vorhanden, der Zustand 1 kann nicht auftreten.
Eine Task, die den Zustand 3, bereit, hat, wird vom Scheduler in Bearbeitung genommen, sobald dies möglich ist. Im einfachsten Fall wird die Bearbeitung solange durchgeführt, bis aus irgendeinem Grund eine Unterbrechung eintritt (kein Interrupt), z. B. auf Daten aus einer anderen Task gewartet werden muß oder die Bearbeitung beendet ist.
Der Zustand 2, angehalten, bedeutet, daß die Task nicht bearbeitet werden darf. Dies ist z. B. der Fall, wenn die letzte Bearbeitung beendet ist und neue Daten abzuwarten sind.
Der Zustand 4, wartend, schließlich wird dazu benutzt, definierte Zeitverzögerungen zu erreichen. Die Task wird erst dann aufgerufen, wenn ein vom System verwalteter Zähler leergezählt ist. Dieser Zähler wird durch eine zum System gehörige Task in festen Zeitintervallen dekrementiert.
Damit eine Task bearbeitet werden kann, muß sie vom Koordinationsprogramm, dem Scheduler, aufgerufen werden. Dies erfolgt nach einer bestimmten Strategie, die im Schedulerprogramm „eingebaut" ist. Die einfachste Aufrufstrategie ist der zyklische Aufruf, auch (amerikanisch) „Round Robin"-Strategie genannt. Dabei werden die Tasks in einer festen Reihenfolge nacheinander abgefragt und aktive Tasks sofort bearbeitet. Bei Unterbrechung einer Task wird jeweils die nächste aktive Task aufgerufen. Nach der letzten Task in der Reihenfolge wird wieder von vorn begonnen.
Der Scheduler hat dabei folgende Aufgaben:

1. Neustart/Wiederstart von aktiven Tasks
2. Umschalten zwischen den Tasks
3. Anhalten von Tasks

Eine wesentliche Vereinfachung des Systemaufbaus ist erreichbar durch einen „kooperativen" Aufbau der einzelnen Tasks. Das heißt, die Tasks sind so aufgebaut, daß sie nie durch langwierige Bearbeitungsfolgen oder durch Warteschleifen die Bearbeitung der anderen Tasks aufhalten. Dafür stellt der Systemkern die nötigen Hilfsmittel zur Verfügung.

Koordination von Tasks

Zusammen mit diesen sind im Systemkern die folgenden Koordinationsfunktionen verfügbar:

1. PAUSE	Unterbrechen einer Task (Rückkehr zum Scheduler)
2. DELAY	Vorübergehendes Stoppen einer Task (Wartend)
3. STOP	Anhalten einer Task
4. ENDE	Beenden einer Task (mit Reinitialisierung)
5. RESTART	Wiederholung einer Task (Neustart von Beginn an)
6. START	Start einer Task

Lediglich die Funktion 6 (Start einer Task) wirkt auf eine andere als die aktuell bearbeitete Task. Alle anderen Funktionen verändern nur den Zustand der aktuellen Task. Ein Aufruf dieser Funk-

tionen (1 bis 5) bewirkt außerdem jeweils die Rückkehr zum Scheduler und den Aufruf aller anderen momentan aktiven Tasks, bevor die aufrufende Task weiter bearbeitet wird.
Wie ist nun ein solches Koordinationsprogramm zu realisieren?
Es ist klar, daß ein direkter Aufruf der Tasks, z. B. über CALL-Befehle, nicht möglich ist, da die Anzahl der aufzurufenden Tasks unbekannt ist und die einzelnen Tasks so wenig wie möglich von der Existenz der übrigen Tasks beeinflußt werden sollen.
Der einfachste und üblicherweise benutzte Weg zum Feststellen der Task-Anzahl besteht darin, die Information, die das System von jeder Task braucht, in einer Tabelle oder Liste, der Task-Zustandsliste, zusammenzufassen.
Der einfachste Weg, den Tasks möglichst viel Unabhängigkeit zu geben, ist, dafür zu sorgen, daß beliebige Unterprogrammaufrufe, also auch Systemaufrufe, „bunt gemischt“ ohne Rücksicht auf die Schachtelungstiefe vorkommen dürfen. Das verlangt aber, jeder Task einen eigenen „privaten“ Stack zuzuordnen. Dort kann eine Task Daten zwischenspeichern, ohne andere Tasks oder den Scheduler zu beeinflussen.

Systementwurf

Im ersten Entwurfsschritt ist nun festzulegen, welche Daten der Scheduler für seine Arbeit benötigt und wie sie organisiert werden. Die größte Flexibilität gestattet die Auslegung als Liste von Datenblöcken, die jeweils die Daten für eine einzelne Task enthalten. Jedes Listenelement enthält einen Zeiger auf das nachfolgende Element, somit können die Taskdaten beliebig im Speicher verteilt sein. Ein Listenelement wird häufig als Task-Leitblock bezeichnet.
Zu jeder Task muß sicher die Startadresse in der Liste eingetragen sein, damit die Task überhaupt gestartet werden kann. Des weiteren muß zu jeder Task der Stackpointer abgelegt werden, um dem Scheduler die Umschaltung der Taskstacks zu ermöglichen.

Die Task-Zustandsliste

Natürlich muß der Scheduler den Zustand der Task kennen. Dafür ist ein Eintrag in der Liste nötig, der im hier beschriebenen System wie folgt aussieht:
Für jede Task ist ein sieben Bit breites Identifikationsfeld vorhanden, das den Start einer Task erlaubt, ohne daß etwas anderes als deren Identifikation bekannt sein muß.
Das Identifikationsfeld ist mit einem Flag zusammengefaßt, das die Zustände „angehalten“ (Flag = 0) und „bereit“ bzw. „wartend“ (Flag = 1) unterscheidet. Die Zustände „bereit“ und „wartend“ werden unterschieden durch den Inhalt eines weiteren Bytes, das gleichzeitig als Wartezeitzähler (DELAY.) dient. Die Wartezeitfelder aller Tasks werden von einer zum System gehörenden Task bearbeitet, die regelmäßig per Interrupt gestartet wird und damit eine Systemzeiteinheit festlegt. Ohne weitere Maßnahmen ist damit eine Verzögerung um max. 255 Systemzeiteinheiten möglich.
Für unser Minimalsystem sieht ein Element der Taskzustandsliste damit also folgendermaßen aus:

1. Wort: (Kalt-)Startadresse der Task
2. Wort: aktueller Stackpointer der Task, bei Initialisierung: Endadresse des Taskstacks
3. Wort: 1. Byte: Task-Identifikation in Bits 0...6, aktiv/inaktiv-Flag in Bit 7
   2. Byte: Verzögerung in Systemzeiteinheiten
4. Wort: Zeiger auf nächstes Taskzustandselement (zyklisch!)

Der „Scheduler“

Nachdem nun die zu bearbeitende Datenstruktur festgelegt ist, kann das Programm entworfen werden, das damit arbeitet. Es gliedert sich in zwei Hauptteile: die Initialisierung und die Task-Aufruf-Schleife. Als Grundstock für ein funktionierendes Steuerprogramm muß unser Scheduler für die Voraussetzungen zum einwandfreien Arbeiten des Systems sorgen.
Als erstes (*Bild 1*) wird deshalb der Stackpointer mit der Anfangsadresse des Scheduler-Stackbereiches geladen. Dann wird die Task-Zustandsliste ins RAM kopiert. Dies ist nötig, weil ihre Elemente während des Programmablaufes verändert werden, was im EPROM nicht möglich wäre. Der ausgeführte Scheduler verwendet das Z80-Register IX als Basisregister für alle Operationen mit den Taskzuständen, deshalb muß es in dieser Programmumgebung (Kontext) immer auf einen gültigen Eintrag der Task-Zustandsliste zeigen. Es wird deshalb zunächst mit der Anfangsadresse

```
.MAIN. - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Initialisierung, .MAIN.-Modul!
                          .EXTERN TASKLISTE, TASKAREA, .TASKLISTE
                          .EXTERN USR.INIT, IR.INIT, SCHEDULER
                          .INTERN RESET, S.INIT

                          ; Initialisierung der Taskzustands-Information
                          ; transportiert alle Taskzustandselemente ins RAM
                          ; (die Werte im ROM muessen bereits die richtigen Links
                          ;  enthalten und bei AKTIV. und DELAY. richtig gesetzt sein!),
                          ; initialisiert die Taskstartadressen und
                          ; setzt den Scheduler-Stackpointer

0000'                S.INIT:
0000'  31 0040"             LXI    SP, SCH.STACK       ; Stackpointer setzen
0003'  21 0000:04           LXI    H, TASKLISTE        ; Liste der vorbesetzten
0006'  11 0000:05           LXI    D, TASKAREA         ; Taskzustandselemente
0009'  01 0000:06           LXI    B, .TASKLISTE       ; nach TASKAREA im RAM
000C'  EDB0                 LDIR                       ; kopieren
000E'  DD21 0000:05         LXI    X, TASKAREA         ; Basisadresse fuer SCHEDULER!
0012'  DD4604               MOV    B, TASK.ID (X)      ; Id der 1. Task merken
0015'                ..NEXT:
0015'  DD5E02               MOV    E, TASK.SP (X)      ; Stackpointer in DE laden
0018'  DD5603               MOV    D, TASK.SP+ 1 (X)   ; DE laden
001B'  CD 003A'             CALL   RESET               ; Startadresse setzen
001E'  DD5E06               MOV    E, NEXT.T (X)       ; Taskzustandselement-
0021'  DD5607               MOV    D, NEXT.T+ 1 (X)    ; adresse ...
0024'  D5                   PUSH   D
0025'  DDE1                 POP    X                   ; ... weiterschalten
0027'  DD7E04               MOV    A, TASK.ID (X)
002A'  B8                   CMP    B                   ; schon einmal rum?
002B'  20E8                 JR     NZ, ..NEXT          ; nochmal
002D'  DDE5                 PUSH   X                   ; Task-Zustands-Zeiger aufheben
002F'  CD 0000:07           CALL   USR.INIT            ; Anwendungs-Initialisierung
0032'  CD 0000:08           CALL   IR.INIT             ; InterRupt-INITialisierung
0035'  DDE1                 POP    X                   ; Zeiger zurueck!
0037'  C3 0000:09           JP     SCHEDULER           ; und los geht's

                     ; Hilfsroutine zum Setzen der (Kalt) Startadresse der aktuellen Task
                     ; als Aufrufadresse fuer den SCHEDULER am Taskstack

003A'                RESET:  ; zerstoert IY und Akku!
                     ; PARAMETER:    Task-Stackpointer in DE
003A'  D5                   PUSH   D                   ; Task-Stackpointer
003B'  FDE1                 POP    Y                   ; nach IY
003D'  DD7E00               MOV    A, 0 (X)            ; Task-Startadresse
0040'  FD7700               MOV    0 (Y), A            ; auf Task-Stack
0043'  DD7E01               MOV    A, 1 (X)            ; als RETurnadresse
0046'  FD7701               MOV    1 (Y), A            ; ablegen
0049'  C9                   RET
```

Bild 1. Initialisierung einer Task-Verwaltung

des Listenspeicherbereichs (im RAM) geladen. Bevor jedoch der erste Aufruf einer Task ausgeführt werden kann, müssen erst noch alle „privaten“ Stacks der Tasks mit deren Kaltstartadressen vorbesetzt werden, da der eigentliche Aufruf per RET geschieht. Die Programmschleife, die das macht, hangelt sich mit Hilfe der Zeiger durch die ganze Taskzustandsliste. In der Schleife wird die Task-Identifikation jedesmal mit der Identifikation der ersten bearbeiteten Task verglichen; weil die Liste zyklisch aufgebaut ist, muß das Programm einmal wieder auf diesen Eintrag stoßen. Damit sind alle Einträge in der Liste bearbeitet. Die Scheduler-Initialisierung ist damit beendet. Normalerweise sind noch weitere vom jeweiligen Einsatzfall abhängige Vorbereitungen auszuführen, die von der Routine USR.INIT ausgeführt werden.
Bevor wir schließlich voll einsteigen können, müssen noch die Ein-/Ausgabeports eingestellt werden und die Interruptbearbeitung muß vorbereitet und freigegeben werden (*Bild 2*). Nach diesen Vorbereitungen ist jetzt endlich alles

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
InterRupt-INITialisierung

                          .EXTERN IRVTABLE

0000'                     IR.INIT:

                                ; Interrupt-Vektoren und Ports initialisieren

0000'   21 0000:04              LXI     H, IRVTABLE           ; Adresse der Vektortabelle laden
0003'   7C                      MOV     A, H
0004'   ED47                    STAI                          ; High Byte -> IR-Basis-register
0006'                     ..SETUP:
0006'   21 0019'                LXI     H, INITABLE           ; Tabelle der Portinitalisierungen


0009'   46                      MOV     B, M                  ; Anzahl der Eintraege
000A'   23                      INX     H
000B'                     ..SCHLEIFE:
000B'   C5                      PUSH    B                     ; Anzahl festhalten
000C'   4E                      MOV     C, M                  ; Port und ...
000D'   23                      INX     H
000E'   46                      MOV     B, M                  ; ... Byteanzahl laden
000F'   23                      INX     H
0010'   EDB3                    OUTIR                         ; Rest geht nach draussen
0012'   C1                      POP     B                     ; Zaehler zurueck
0013'   10F6                    DJNZ    ..SCHLEIFE            ; so oft wie angegeben
0015'   ED5E                    IM2                           ; Interrupt spezifizieren
0017'   FB                      EI                            ; und freigeben
0018'   C9                      RET                           ; fertig

                          ; Festdaten:   Port-Initialisierungswerte (Beispielwerte)

0019'                     INITABLE:
0019'   09                .BYTE   9                           ; Anzahl der SETUP-Definitionen

001A'   020508CFFFB7      .BYTE  ANZEIGE+ CTRL,  5, 8, 11001111B, 11111111B, 10110111B, 00000000B
0021'   030508CFFFB7      .BYTE  PROGRAMM+ CTRL, 5, 8, 11001111B, 11111111B, 10110111B, 01111111B
0028'   0C030A4F83        .BYTE  LOWTEMP+ CTRL,  3,10, 01001111B, 10000011B
002D'   0D030A4F03        .BYTE  HIGHTEMP+ CTRL, 3,10, 01001111B, 00000011B
0032'   1702A90D          .BYTE  PPICTRL,        2, 10101001B, (6 < 1 ! 1)
0036'   1E0300077D        .BYTE  CTC0,           3, 0, 00000111B, 125.
003B'   1F02C7FA          .BYTE  CTC0+ 1,        2, 11000111B, 250.
003F'   2002C7F0          .BYTE  CTC0+ 2,        2, 11000111B, 240.
0043'   210143            .BYTE  CTC0+ 3,        1, 01000011B    ; wird nicht gebraucht
```

Bild 2. Initialisierung der Ports und Interrupt-Vektoren

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin

                     .EXTERN TASKAREA              ; RAM-Bereich der Taskzustandsliste
                     .EXTERN RESET
                     .INTERN SAVESP
                     .INTERN PAUSE, DELAY, STOP, END, RESTART, START

                     ; Datenstruktur fuer Taskzustandselemente

0007                 AKTIV.   == 7                 ; Flag-Bit
007F                 ID.MASK == 7FH                ; Maske fuer Taskioentifikation
0002                 TASK.SP == 2                  ; Offset in Taskzustand
0004                 TASK.ID == 4                  ;       "
0005                 DELAY.  == 5                  ;       "
0006                 NEXT.T  == 6                  ;       "

                     ; Ein Taskzustandselement hat folgenden Aufbau:
                     ; 1. Wort: (Kalt-) Startadresse der Task
                     ; 2. Wort: aktueller Stackpointer der Task,
                     ;          bei Initialisierung: Endadresse des Taskstacks
                     ; 3. Wort:
                     ;       1. Byte: Taskidentifikation in Bits 0..6,
                     ;                aktiv/inaktiv-Flag in Bit 7
                     ;       2. Byte: Verzoegerung in Systemzeiteinheiten
                     ; 4. Wort: Zeiger auf naechstes Taskzustandselement (zyklisch!)


0000'                SCHEDULER:
                     ; Eingang: IX mit Adresse eines Taskzustandselementes besetzt
                     ;          (Taskzustandselement muss richtig besetzt sein!)

                     ; Testen, ob Task aktiv
0000'   DDCB047E           BIT     AKTIV., TASK.ID (X)
0004'   2814               JR      Z, NEXTTASK

                     ; Testen, ob Task wartend
0006'   DD7E05             MOV     A, DELAY. (X)
0009'   B7                 ORA     A
000A'   200E               JR      NZ, NEXTTASK

                     ; Task aufrufen
000C'   DDE5               PUSH    X
000E'   ED73 0000"         SSPD    SAVESP                 ; Speicherplatz fuer Stackpointer
0012'   DD6E02             MOV     L, TASK.SP (X)
0015'   DD6603             MOV     H, TASK.SP+ 1 (X)      ; Task-Stackpointer laden
0018'   F9                 SPHL                           ; Stack umschalten
0019'   C9                 RET                            ; Bearbeitung aufnehmen

                     ; Rueckkehr aus der Task:     je nach Bearbeitungszustand ueber
                     ;                             verschiedene Eintrittspunkte

                     ; Fortschalten auf naechste Task
001A'                NEXTTASK:      ; durch Pointer weiterhangeln
001A'   DD5E06             MOV     E, NEXT.T (X)          ; Taskzustandselement-
001D'   DD5607             MOV     D, NEXT.T+ 1 (X)       ; adresse nach
0020'   D5                 PUSH    D
0021'   DDE1               POP     X                      ; Indexregister
0023'   18DB               JR      SCHEDULER              ; weiter mit diesem

                     ; Routine zur Kontext-Umschaltung
0025'                SWITCH:
0025'   FDE1               POP     Y                      ; Rueckkehradresse holen
0027'   2A 0000"           LHLD    SAVESP                 ; Scheduler-Stackpointer retten
002A'   ED73 0000"         SSPD    SAVESP                 ; Task-Stackpointer
002E'   ED5B 0000"         LDED    SAVESP                 ; zwischenspeichern
0032'   F9                 SPHL                           ; Stack umschalten
0033'   DDE1               POP     X                      ; Taskelementadresse zurueck
0035'   DD7302             MOV     TASK.SP (X), E         ; Task-Stackpointer
0038'   DD7203             MOV     TASK.SP+ 1 (X), D      ; ablegen
003B'   FDE9               PCIY                           ; fertig, a la RETurn weiter

                     ; Von Anwendungsprogramm aufrufbare Eintrittspunkte:

003D'                PAUSE:  ; Rueckgabe an Scheduler, wenn Warten noetig
                             ; keine Parameter, Rueckkehr wie aus Subroutine
                             ; alle benoetigten Register sind zu retten!
003D'   CD 0025'           CALL    SWITCH
0040'   18D8               JR      NEXTTASK

0042'                DELAY:  ; wie PAUSE, aber fuer spezifizierte Zeit
                     ; PARAMETER:    Verzoegerung in Akku
0042'   CD 0025'           CALL    SWITCH
0045'   DD7705             MOV     DELAY. (X), A          ; Verzoegerung ablegen
0048'   18D0               JR      NEXTTASK

004A'                STOP:   ; Task inaktivieren, z.B. fuer Intertask-kommunikation
                             ; sonst wie PAUSE
004A'   CD 0025'           CALL    SWITCH
004D'   DDCB04BE           RES     AKTIV., TASK.ID (X)    ; Task inaktiv setzen
0051'   18C7               JR      NEXTTASK

0053'                END:    ; Beenden einer Task mit Reinitialisierung der Startadresse
                             ; Stack  m u s s  leer sein !!!!!
0053'   CD 0025'           CALL    SWITCH                 ; Task-Stackpointer noch in DE
0056'   DDCB04BE           RES     AKTIV., TASK.ID (X)    ; Task inaktiv setzen
005A'   CD 0000:05         CALL    RESET                  ; Eintrittsadresse reinitialisieren
005D'   18BB               JR      NEXTTASK

005F'                RESTART:; Wiederstart einer Task von Anfang an, evtl. mit Verzoegerung
                             ; Stack  m u s s  leer sein !!!!!
                     ; PARAMETER:    Verzoegerung in Akku
005F'   CD 0025'           CALL    SWITCH
0062'   DD7705             MOV     DELAY. (X), A          ; Verzoegerung ablegen
0065'   CD 0000:05         CALL    RESET                  ; Eintrittsadresse reinitialisieren
0068'   18B0               JR      NEXTTASK
```

Bild 3. Routinen zur Verwaltung der Tasks

getan, um die reguläre Programmbearbeitung aufnehmen zu können (*Bild 3*). Das Basiszeigerregister IX zeigt auf das erste Element der Task-Zustandsliste. Im ersten Schritt wird nun geprüft, ob die aktuelle Task angehalten ist. Wenn dies der Fall ist, wird sofort zur nächsten Task weitergegangen. Ansonsten wird im zweiten Schritt geprüft, ob der Wartezeitzähler läuft. Nur wenn dieser auf Null steht, ist die Task aktiv und kann aufgerufen werden.

Dazu wird zuerst das Basiszeigerregister auf den Scheduler-Stack gerettet. Der momentane Stackpointerinhalt wird dann an einer dafür reservierten Stelle abgespeichert. Jetzt kann das Stackpointerregister mit dem Task-Stackpointer geladen werden. Im Task-Stack steht als letzter Eintrag die Adresse des nächsten zu bearbeitenden Befehls der Task (am Anfang steht dort die Startadresse der Task). Ein einfaches RET startet jetzt die Bearbeitung der Task.
Eine Rückkehr zum Scheduler findet mit jedem Aufruf einer Systemroutine in der laufenden Task (mit Ausnahme der Start-Routine) statt. Dabei kommt ein Trick zur Auswirkung: der CALL-Befehl beim Aufruf der Systemroutine hat automatisch die nächste Befehlsadresse der Task auf deren Stack abgelegt. Die Schedulerroutine SWITCH, die bei der Rückkehr immer als erstes aufgerufen wird, „friert“ diesen Zustand des Taskstacks ein und führt eine Kontextumschaltung in den Scheduler-Kontext durch (d. h. die gesamte Programmumgebung des Schedulers, die auch den Stack beinhaltet, wird zugänglich gemacht). Dazu holt SWITCH zunächst die überflüssige eigene Rückkehradresse vom Stack in IY. Dann lädt sie – einigermaßen umständlich – den Task-Stackpointerwert in das Registerpaar DE und schaltet das Stackpointer-Register auf den Schedulerstack um. Von diesem wird das Basiszeigerregister zurückgeholt und damit ist der letzte Task-Zustand wieder zugänglich. Dort wird nun der aktuelle Task-Stackpointerwert abgelegt. Über die in IY stehende Rückkehradresse wird danach die Bearbeitung im Systemkontext fortgesetzt.
Das damit erfolgte „Einfrieren“ des Taskstacks erlaubt, jede beliebige andere Aktivität zwischendurch auszuführen, ohne den Kontext (die Umgebung) der gerade verlassenen Task zu stören. In unserem Fall werden nun erst einmal alle anderen Tasks geprüft und, wenn aktiv, aufgerufen, bis nach einem vollständigen Umlauf die eben verlassene Task wieder an die Reihe kommt.

Doch weiter der Reihe nach:
Nach der Rückkehr von SWITCH wird evtl. noch der Zustand der letzten Task verändert, z. B. eine Wartezeit eingetragen, dann schaltet NEXTTASK das IX-Register zum nächsten Element der Task-Zustandsliste weiter. Damit wird dann genauso verfahren wie gerade beschrieben.

Der geschilderte Ablauf zeigt auch, warum die einzelnen Tasks „kooperativ“ programmiert werden müssen. Da der Scheduler keine Eingriffsmöglichkeit in eine laufende Task hat, kann eine Schleife, in der keine Systemroutine aufgerufen wird, den ganzen Ablauf anhalten. Deshalb sind Warteschleifen nicht erlaubt!

Neustart einer Task

Der bis hier behandelte Teil des Systems erlaubt bereits den quasi-parallelen Ablauf mehrerer Programme durch verschachtelte Abarbeitung. Jedoch gibt es damit keine Möglichkeit, eine angehaltene Task zum Laufen zu bringen.

Dazu ist ein Zugriff auf den Taskzustand einer anderen als der aktuellen Task nötig. Dies leistet die Prozedur START (*Bild 4*). Ein direkter Zugriff auf einen bestimmten Eintrag der Task-Zustandsliste ist normalerweise nicht möglich, weil die Reihenfolge der Einträge und evtl. sogar ihre Lage im Speicher nicht bekannt sind. Wir müssen also mit einem bekannten Eintrag anfangen und die Liste anhand der Zeigerverkettung verfolgen, bis der gesuchte Eintrag gefunden ist. Die Vorgehensweise ist bereits bekannt: sie ist dieselbe wie bei der Stack-Initialisierung. Lediglich der Vergleich der Task-Identifikation muß verfeinert werden: die Identifikation muß unabhängig vom Zustand des höchstwertigen (Flag-)Bits gefunden werden (es ist aber auch die Beschränkung der Suche nur auf inaktive Tasks möglich.

Wenn die gesuchte Task bereits aktiv oder wartend ist, wird sie nicht beeinflußt; die Suchstrategie ist dann entsprechend anzupassen). Zusätzlich ist in der gezeigten Routine noch eine Sicherung eingebaut: wenn die gesuchte Task-Identifikation nicht vorhanden sein sollte, bricht die einfache Suchschleife nicht ab und das System „hängt sich auf“. Das wird verhindert, indem geprüft wird, ob die Liste bereits einmal ganz abgesucht worden ist. In diesem Fall wird die Suche abgebrochen und über das Carry-Flag Fehlanzeige zurückgemeldet. Außerdem zeigt das Zero-Flag an, ob die gestartete Task wirklich angehalten war. Mit dieser Routine können wir nun eine inaktive Task „aufwecken“, und eine Task kann sich am Ende ihrer Aktivitä-

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin

 006A'               START:  ; Start einer Task, evtl. mit Verzoegerung
                     ; PARAMETER:    Verzoegerung in Akku
                     ;               Taskidentifikation in C-Register
                     ; ERGEBNIS:     Z  gesetzt, wenn Task inaktiv war
                     ;               CY gesetzt, wenn Task nicht gefunden
 006A'  FDE5                 PUSH    Y
 006C'  D5                   PUSH    D
 006D'  C5                   PUSH    B
 006E'  F5                   PUSH    PSW                 ; Register retten
 006F'  FD21 0000:04         LXI     Y, TASKAREA         ; Scheduler-Kontext verfuegbar machen
 0073'  3E7F                 MVI     A, ID.MASK
 0075'  FDA604               ANA     TASK.ID (Y)         ; Taskidentifikation holen
 0078'  47                   MOV     B, A                ; erste merken
 0079'               ..SUCHE:
 0079'  B9                   CMP     C                   ; mit gesuchter vergleichen
 007A'  2818                 JR      Z, ..DIESE          ; gefunden?
 007C'  FD5E06               MOV     E, NEXT.T (Y)       ; naechstes Taskzustands-
 007F'  FD5607               MOV     D, NEXT.T+ 1 (Y)    ; element ...
 0082'  D5                   PUSH    D
 0083'  FDE1                 POP     Y                   ; .... ansteuern
 0085'  3E7F                 MVI     A, ID.MASK
 0087'  FDA604               ANA     TASK.ID (Y)         ; naechste Taskid holen
 008A'  B8                   CMP     B                   ; wieder am Anfang?
 008B'  20EC                 JR      NZ, ..SUCHE         ; weitersuchen
 008D'  F1                   POP     PSW
 008E'  C1                   POP     B
 008F'  D1                   POP     D
 0090'  FDE1                 POP     Y
 0092'  37                   SCF                         ; melde nicht gefunden
 0093'  C9                   RET

 0094'               ..DIESE:
 0094'  F1                   POP     PSW
 0095'  FDCB047E             BIT     AKTIV., TASK.ID (Y) ; melde (in)aktiv/wartend
 0099'  FDCB04FE             SET     AKTIV., TASK.ID (Y) ; setze aktiv/wartend
 009D'  FD7705               MOV     DELAY. (Y), A       ; setze Verzoegerung
 00A0'  C1                   POP     B                   ; hole restliche Register zurueck
 00A1'  D1                   POP     D
 00A2'  FDE1                 POP     Y
 00A4'  C9                   RET
```

Bild 4. Routine zum Start einer inaktiven Task

ten ruhig „schlafenlegen“ (anhalten), bis sie wieder gebraucht wird.

Das Aktivieren einer Task kann von beliebiger Stelle aus geschehen. Es muß lediglich die Adresse von START und die Identifikation der zu startenden Task bekannt sein. Der Start kann auch von einer Interruptroutine ausgeführt werden. Die Interruptroutine bleibt dadurch kurz und belastet das System nicht. Die Hauptarbeit – die für eine Interruptroutine zu komplex sein kann – wird dann im normalen Taskmodus ausgeführt. Ein Beispiel für eine solche Task ist die zum System gehörige Wartezeitbearbeitung (TICK, *Bild 5*).

Die Interruptroutine IR.TICK startet lediglich die Task TICK; erst diese dekrementiert die Wartezeitzähler aller Tasks. Die Bearbeitung läuft wieder nach dem Schema der Stackinitialisierung ab. Wie dort wird die gesamte Liste der Taskzustände einmal durchlaufen, und bei jeder Task wird der DELAY-Zähler dekrementiert, wenn er nicht bereits den Wert Null hat. Da das nicht im Interruptmodus geschieht, kann es keinen Konflikt mit dem Setzen oder Löschen des Zählers von anderer Stelle (START) geben. TICK zeigt auch den Aufruf einer Systemroutine. Der Aufruf von ENDE gibt die Bearbeitung wieder an den Scheduler zurück und setzt als Rückkehradresse für den nächsten Taskaufruf die Startadresse von TICK ein. Damit fängt TICK bei jedem Aufruf von vorn an und bearbeitet die gesamte Task-Zustandsliste.

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin - Beispiel-Task DRUCK

0003                   $READY  == 3                          ; READY-Bit Drucker-Steuerung

                       .EXTERN DRUCKZEIGER
0000'                  DRUCK:
0000'   DB22                   IN     READY
0002'   CB5F                   BIT    $READY, A
0004'   280F                   JR     Z, ..WARTE             ; Drucker nicht bereit?
0006'   2A 0000"               LHLD   DRUCKZEIGER
0009'   7E                     MOV    A, M
000A'   FEFF                   CPI    $END                   ; Textende testen
000C'   CC 0000:04             CALL   Z, ENDE                ; fertig?
000F'   23                     INX    H
0010'   22 0000"               SHLD   DRUCKZEIGER            ; Druckzeiger weiterstellen
0013'   D320                   OUT    DRUCKER                ; Zeichen ausgeben
0015'                  ..WARTE:
0015'   AF                     CALL   RESTART                ; eine Runde aussetzen
0016'   CD 0000:05

                       ; Hilfsroutine, die signalisiert, ob Drucktask verfuegbar ist

0019'                  BEREIT: ; gibt erst zurueck, wenn Drucktask frei ist
                               ; zerstoert alle Register!
0019'   3EFF                   MVI    A, $END
001B'   2A 0000"               LHLD   DRUCKZEIGER
001E'   BE                     CMP    M
001F'   C8                     RET    Z
0020'   CD 0000:06             CALL   PAUSE                  ; eine Runde aussetzen
0023'   18F4                   JR     BEREIT                 ; wieder probieren
```

Bild 6. Beispiel aus einem Protokolldrucker

Ein Anwendungsbeispiel

Schließlich soll noch ein Beispiel aus einer Anwendung des vorgestellten Systems gezeigt werden (*Bild 6*, vgl. [2]). Es handelt sich hier um eine Task, die einen von einer anderen Task vorbereiteten Text auf einem Drucker ausgibt. Der Grundaufbau der Task wird zumindest denen nicht fremd sein, die sich schon einmal mit Schnittstellen und deren Programmierung befaßt haben. Hier sind jedoch zwei Stellen besonders zu beachten:
Es gibt keine echte Warteschleife für den Fall, daß der Port noch keine neuen Daten aufnehmen kann, da das System nicht blockiert werden darf. Statt dessen wird die Sytemfunktion RESTART aufgerufen, die zum Scheduler – und damit zu den anderen Tasks – zurückschaltet und außerdem dafür sorgt, daß beim nächsten Aufruf die Task bei ihrer Startadresse wieder aufgenommen wird. Damit kommen bei jeder vergeblichen Portabfrage alle anderen aktiven Tasks zur Bearbeitung.
Dasselbe geschieht, wenn ein Zeichen ausgegeben wurde, da dann der Drucker erst einmal beschäftigt ist bzw. das Zeichen übertragen werden muß.
Wenn jedoch das letzte Zeichen des Textes ausgegeben ist und der Druckzeiger auf eine Endemarke zeigt, beendet sich das Druckprogramm durch den Aufruf von ENDE und wartet inaktiv auf den nächsten Start (wie TICK). So, wie die letzte Abfrage hier programmiert ist, würde in einem Einprogrammsystem Zeit verschenkt. Im vorliegenden Fall ist dies aber bedeutungslos, da ja ein Warten auf den Port die anderen Tasks nicht am Ablaufen hindert. Die gezeigte Form spart dagegen etwas Programmspeicher, da der Vergleich gleich nach dem sowieso nötigen Holen des Ausgabezeichens steht.

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin
                       ; Task-Identifikation:
 0054                  $TICK   == 'T'

 0034'                 IR.DELAY:      ; Verzoegerungsbearbeitung (Systemzeit)
 0034'  08                     EXAF
 0035'  D9                     EXX
 0036'  0E54                   MVI    C, $TICK
 0038'  AF                     XRA    A
 0039'  CD 0000:05             CALL   START
 003C'  D9                     EXX
 003D'  08                     EXAF
 003E'  FB                     EI
 003F'  ED4D                   RETI
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin

                       .EXTERN TASKAREA                      ; RAM-Bereich der Taskzustandsliste

 0000'                 TICK:   ; dekrementiert Verzoegerungszaehler <> 0 aller Tasks
 0000'  FD21 0000:04           LXI    Y, TASKAREA            ; Scheduler-Kontext verfuegbar machen
 0004'  3E7F                   MVI    A, ID.MASK
 0006'  FDA604                 ANA    TASK.ID (Y)            ; Taskid holen
 0009'  47                     MOV    B, A                   ; erste merken
 000A'                 ..SCAN:
 000A'  FD7E05                 MOV    A, DELAY. (Y)          ; Verzoegerung holen
 000D'  B7                     ORA    A
 000E'  2803                   JR     Z, ..NULL              ; = 0 ?
 0010'  FD3505                 DCR    DELAY. (Y)             ; dekrementieren
 0013'                 ..NULL:
 0013'  FD5E06                 MOV    E, NEXT.T (Y)          ; naechstes Taskzustands-
 0016'  FD5607                 MOV    D, NEXT.T+ 1 (Y)       ; element ...
 0019'  D5                     PUSH   D
 001A'  FDE1                   POP    Y                      ; ... ansteuern
 001C'  3E7F                   MVI    A, ID.MASK
 001E'  FDA604                 ANA    TASK.ID (Y)            ; naechste Taskid holen
 0021'  B8                     CMP    B                      ; einmal rum?
 0022'  CC 0000:05             CALL   Z, ENDE                ; dann fertig
 0025'  18E3                   JR     ..SCAN                 ; sonst weiterarbeiten
```

Bild 5. Wartzeit-Bearbeitung mit der Task TICK

In Bild 6 ist zusätzlich zur eigentlichen Drucktask noch eine Dienstprozedur (BEREIT) angegeben, die anderen Tasks zur Verfügung steht. Durch einen Aufruf von BEREIT kann eine Task ihren Ablauf mit der Druckausgabe synchronisieren. Ein solcher „Export“ von Funktionen fördert ebenfalls die Übersichtlichkeit eines Programms, da alle Funktionen, die ein Modul betreffen, in diesem selbst ausgeführt werden und ein „fremdes“ Modul keine Kenntnis über deren Aufbau braucht. Die hier gezeigte Routine BEREIT entspricht in ihrem Aufbau völlig einer einfachen Warteschleife in einem Einprogrammsystem. Da aber einfache Warteschleifen in unserem Multitaskingsystem verboten sind – sie sind nicht kooperativ – muß innerhalb der Schleife mindestens einmal eine Systemfunktion aufgerufen werden, die den Ablauf anderer Tasks erlaubt. Im Beispiel leistet dies die Prozedur PAUSE. Diese Routine ist, im Unterschied zu RESTART und ENDE, genauso zu verwenden wie eine beliebige andere Subroutine. D. h. die Bearbeitung der Task geht wie bei einer solchen mit dem dem Aufruf folgenden Befehl weiter, ohne daß die Programmumgebung verändert ist. Die Wirkung der Systemroutinen ist im Bild 4 bei den einzelnen Routinen selbst angegeben.

Für viele Zwecke ausreichend

Mit dem vorgestellten „Micro-Multitasking-Scheduler“ können bereits mit kleinsten Konfigurationen die Programmierverfahren der Prozeßrechentechnik verwendet werden. Es fehlen natürlich noch viele Möglichkeiten der großen Systeme. So ist hier keine Unterstützung von Ein-/Ausgabevorgängen auf Systemebene vorhanden, und auch der Datenaustausch zwischen den Tasks wird nicht unterstützt. Aber bei Computern der Ausbaustufe des Z80-EMUF sind die Programme wohl noch einfach genug, daß zur Organisation dieser Vorgänge keine Klimmzüge nötig sind. Außerdem kann man den hier vorgestellten Kern ja noch um die zusätzlich gebrauchten Funktionen erweitern.

Mit dem vorhandenen Systemkern wurde ein Protokolldrucker ausgeführt, der interruptgesteuert ein Dutzend Eingänge überwacht, ihre Zustände – teils in festen Zeitintervallen – zusammen mit einem Meßwert protokolliert und das Protokoll mit Datum, Uhrzeit und diversen anderen Zusatzinformationen versieht. Datum und Uhrzeit werden ebenfalls interruptgesteuert intern geführt und können mit einer eingebauten Tastatur überprüft und eingestellt werden. Insgesamt enthält das Betriebsprogramm sechs Tasks und ist in knapp als 2 KByte Programmspeicher untergebracht. Der Systemkern benötigt davon etwa 300 Byte.

Die Listings wurden mit einem Z80-Assembler erstellt, dessen Befehlssyntax weitgehend der von Intel für den 8080 eingeführten entlehnt ist. Lediglich die Erweiterungen des Z80-Befehlssatzes ohne 8080-Entsprechungen sind der Zilog-Spezifikation entnommen.

Der Assembler erstellt relokatiblen (verschiebbaren) und linkfähigen (bindefähigen) Code. Er ermöglicht damit ein einfaches Aufspalten des Quellenprogramms in Module durch Angabe von internen Symbolen (die in anderen Modulen verwendbar sein sollen) und von externen Symbolen, die in anderen Modulen definiert sind.

Literatur

z1] *Kanis, W.*: Der Z80-EMUF. mc 1983, Heft 4, Seite 112.

[2] *Gaulke, E.*: Z80-EMUF als Spooler. mc 1983, Heft 10, Seite 98.

Dr. Dieter Götz

Z80-EMUF mißt Spannung und pH-Wert

Ein erweiterter Z80-EMUF [1] wird mit Hilfe eines zusätzlichen A/D-Wandlers und etwas Software zu einem Millivolt- bzw. pH-Meter.

Als A/D-Wandler wird ein 12-Bit-CMOS-A/D-Wandler von Intersil (ICL-7109) verwendet.

Es handelt sich hierbei um einen ausgesprochenen komfortablen Wandler. Er arbeitet nach der „Dual-Slope"-Integrationsmethode und besitzt TTL-kompatible Tri-State-Ausgänge. Zur Steuerung der A/D-Wandlung gibt es einen RUN/$\overline{\text{HOLD}}$-Eingang. Liegt dieser Ausgang auf logisch 1 bzw. bleibt er unbeschaltet, führt der Baustein eine Wandlung nach der anderen aus. Wird er auf logisch 0 gelegt, so beginnt die Wandlung erst beim Umschalten auf logisch 1. Eine zweite Möglichkeit zur Steuerung bietet der Zustand des Statusbits. Während der Wandlung liegt es auf logisch 1, am Ende geht es auf logisch 0. Im vorliegenden Fall wird der RUN/$\overline{\text{HOLD}}$-Eingang unbeschaltet gelassen und die Wandlung erfolgt softwaremäßig mit Hilfe des Statusbits.

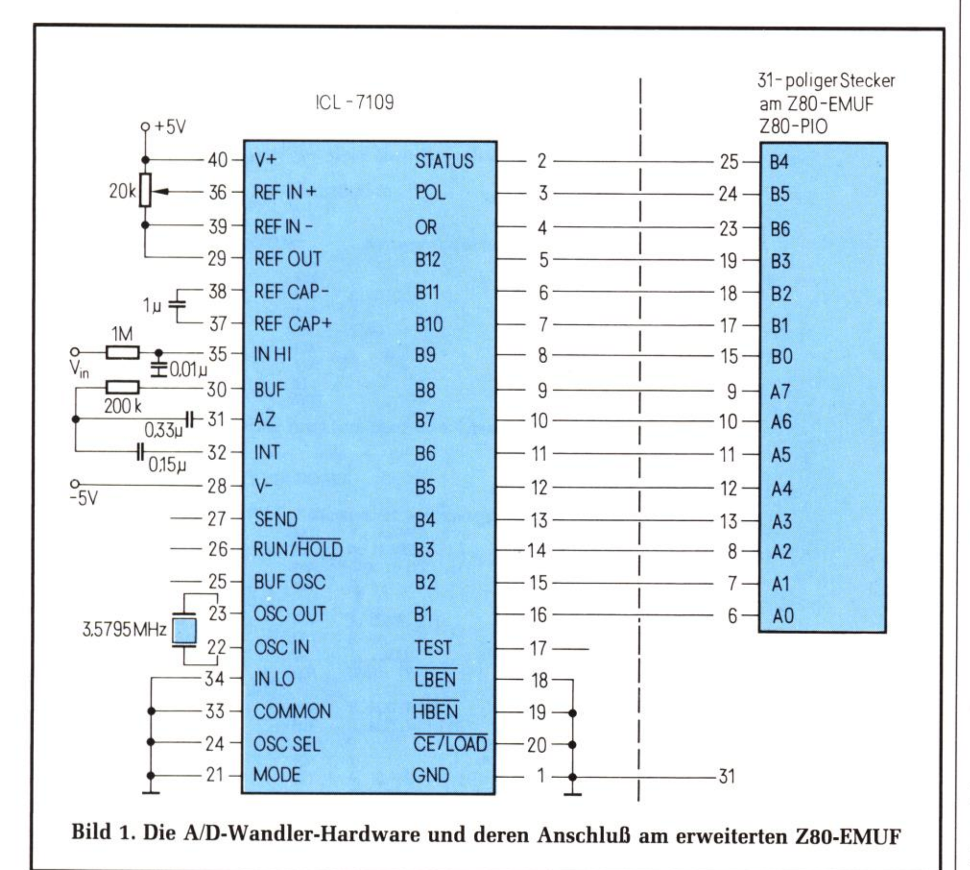


Bild 1. Die A/D-Wandler-Hardware und deren Anschluß am erweiterten Z80-EMUF

Außerdem besitzt der Wandler noch einen Ausgang zur Anzeige der Meßbereichsüberschreitung und der Polarität.

Der ICL 7109 hat die Möglichkeit zur Wahl zweier Betriebsarten: Handshakemodus oder Direktmodus. Es wurde der Direktmodus gewählt. Als Versorgungsspannungen werden ± 5 V benötigt. Die positive Versorgungsspannung kann z. B. an Pin 27 der 31poligen Leiste des Z80-EMUF abgegriffen werden.

Wenig Hardware mit viel Leistung

Wie man aus *Bild 1* sieht, sind neben einem Schwingquarz lediglich einige Widerstände und Kondensatoren notwendig. Die Schaltung wurde so bemessen, daß sie einen Meßbereich von ± 5 V umfaßt. Die genaue Einstellung erfolgt mit dem 20-kΩ-Potentiometer.

Die digitalen Ausgänge (Bit 1...Bit 12, Status, Overrange, Polarität) werden direkt auf die entsprechenden Pins des 31poligen Leistensteckers des EMUF gelegt.

EMUF als Digital-Millivoltmeter

Bild 2 zeigt das Maschinenprogramm, das den EMUF zum Millivoltmeter macht. Es beginnt mit der RAM-Adresse 8200H und umfaßt etwa 1 KByte. Von dem Monitorprogramm [1] werden zwei Unterprogramme verwendet, nämlich Tastaturabfrage/Eingabe (0097H) und Zeichenausgabe auf dem Display (00BDH). Das Programm umfaßt die Lösung folgender Aufgaben:

- □ Auswertung der A/D-Wandlung
- □ Hexadezimal/Dezimal-Umwandlung
- □ Umrechnen der eingelesenen Werte auf den Meßbereich von −5 V bis +5 V
- □ Ausgabe der angelegten Meßspannung in mV auf dem Display

Das Programm wird durch Eingabe der Adresse 8200H und anschließendes Drücken der Go-Taste gestartet. Durch Drücken der Display-Taste kann der in diesem Augenblick im Display erscheinende Wert fixiert werden. Durch Drükken der Enter-Taste wird der Programmlauf fortgesetzt. Mit Break gelangt man zurück ins Monitorprogramm.

EMUF als pH-Meter

Zur pH-Messung wird als Elektrode die heute nahezu ausschließlich verwendete

```
82oo: af 32 f9 87 11 fa 87 cd od 82 c3 oo oo d5 3e cf
821o: d3 o2 3e ff d3 o2 3e cf d3 o3 3e ff d3 o3 db o1
822o: e6 4o 28 1o 21 e1 87 3e 4o o6 o6 77 23 1o fc cd
823o: bd oo 18 52 db o1 e6 1o 2o o5 cd bd oo 18 f5 db
824o: oo 6f db o1 e6 of 67 54 5d o6 o2 cd d9 82 19 o6
825o: o3 cd d9 82 af ed 52 o6 o4 cd d9 82 19 d1 eb 73
826o: 23 72 2b eb d5 3a f9 87 b7 28 o2 d1 c9 cd a3 82
827o: cd 9o o2 db o1 e6 2o 21 e5 87 2o o6 3e 4o 77 af
828o: 18 o2 af 77 23 77 cd bd oo cd 97 oo 7b fe o4 2o
829o: o2 d1 c9 fe o5 cc eo 82 db o1 e6 1o 28 8o cd bd
82ao: oo 18 f5 11 e8 o3 cd c3 82 41 11 64 oo cd c3 82
82bo: 78 cd ce 82 11 oa oo cd c3 82 6o 41 4d 78 cd ce
82co: 82 68 c9 af 4f ed 52 38 o3 oc 18 f9 19 c9 cb 27
82do: cb 27 cb 27 cb 27 b1 47 c9 cb 2a cb 1b 1o fa c9
82eo: cd bd oo cd 97 oo 7b fe o6 2o f5 c9 oo oo oo oo
82fo: oo oo oo oo oo oo oo oo oo oo oo oo oo oo oo oo
```

Bild 2. Dieses Programm macht den EMUF zu einem Digital-Millivoltmeter

Glaselektrode eingesetzt (z. B. bei der Firma Ingolf oder der Firma Schott erhältlich). Außerdem benötigt man zum Kalibrieren zwei Lösungen mit genau definierten pH-Werten. Dazu verwendet man zwei Pufferlösungen mit dem pH-Wert 7 und dem pH-Wert 9 (Pufferlösungen sind z. B. bei der Firma Merck, Darmstadt, erhältlich). Da alle späteren pH-Berechnungen sich auf diese beiden Eichpuffer beziehen, sollte die Glaselektrode vor dem Wechseln der Puffer sorgfältig mit destilliertem Wasser abgespült und die Pufferlösungen von Zeit zu Zeit gewechselt werden. *Bild 3* zeigt das Programm zur pH-Messung.

Die Kalibrierung zur pH-Messung

Soll der EMUF als pH-Meter eingesetzt werden, muß auch das Programm nach Bild 2 im RAM stehen, da das pH-Programm auf einige Routinen zurückgreift. Das Programm Kalibrierung beginnt bei der Adresse 8300H und geht bis 834FH. Es wird gestartet durch Eingabe der Adresse 8300H und anschließendes Drücken der Go-Taste. Im Display erscheint dann: PUF1= 7. Die Glaselektrode sollte jetzt am A/D-Wandler angeschlossen sein und in der Pufferlösung mit pH=7 stehen. Nach Drücken der Taste Enter beginnt die A/D-Wandlung und im Display erscheint der entsprechende Meßwert des Eichpuffers 7 in mV. Er liegt in der Nähe von 0 V. Zu beachten ist, daß sich an der Glasmembran der Glaselektrode relativ komplexe chemische Ionenaustauschvorgänge abspielen, so daß die Einstellung des chemischen Gleichgewichts einige Zeit benötigt. Ändert sich der Meßwert im Display nicht mehr, wird die Taste Break gedrückt; dadurch wird der augenblickliche Wert für pH 7 in der Speicherzelle 87FAH abgelegt und im Display erscheint: PUF2= 9. Nach Abspülen der Elektrode wird diese jetzt in den Puffer mit pH 9 gestellt und der Meßvorgang analog zum ersten Puffer wiederholt. Durch Drücken der Taste Break ist der Kalibriervorgang beendet. Die aus den beiden Puffern berechnete Empfindlichkeit der Glaselektrode (Δ mV/pH-Einheit) ist in Speicherzelle 87FEH abgelegt.

An sich genügt es, einmal zu Beginn der pH-Messungen die Kalibrierung durchzuführen. Bei längeren Messungen ändert sich jedoch manchmal die Empfindlichkeit der Elektrode. Eine gelegentliche Nachkalibrierung ist deshalb empfehlenswert. Ist die Empfindlichkeit der Elektrode bekannt, kann sie natürlich auch direkt in 87FEH abgelegt werden. In 87FFH ist dann eine Null einzugeben.

Die pH-Messung

Das Programm für die pH-Messung beginnt ab 8350H. Im Display erscheint zunächst: $\overline{\text{n}}$ESSEn. Nach Drücken von Enter beginnt die pH-Messung. Das Ergebnis erscheint im Display. Es erfolgt eine kontinuierliche pH-Messung; der Rücksprung ins Monitorprogramm erfolgt wieder durch die Break-Taste. Die Bedienung erfolgt analog der Spannungsmessung.

Eingabe des Programms

Zur Eingabe des Programms gibt es drei Möglichkeiten:

- ☐ Per Hand. Da die vorliegenden Programme schon relativ umfangreich sind, ist dies ein wenig mühselig.
- ☐ Wird der EMUF oft als Millivoltmeter und zur pH-Messung eingesetzt, können die Programme neben dem Monitor noch im EPROM 2716 abgespeichert werden.
- ☐ Wie in [1] beschrieben, können über Bit 7 der PIO 0, Port B (Pin 22 des 31poligen Leistensteckers), Daten aus- bzw. eingegeben werden.

```
83oo: 21 o6 83 c3 df 83 3e a8 12 cd bd oo cd 97 oo 7b
831o: fe o6 2o f5 11 fa 87 af 32 f9 87 cd od 82 21 24
832o: 83 c3 df 83 3e f9 12 cd bd oo cd 97 oo 7b fe o6
833o: 2o f5 11 fc 87 cd od 82 ed 5b fa 87 2a fc 87 af
834o: ed 52 cb 3c cb 1d 22 fe 87 oo oo oo oo oo oo oo

835o - 849f: pH-Messung


835o: 21 e6 87 3e 6c 77 2b 3e 75 77 2b 3e 7b 77 2b 77
836o: 3e 75 2b 77 3e 4c 2b 77 cd bd oo cd 97 oo 7b fe
837o: o6 2o f5 cd 81 83 cd 97 oo 7b fe o4 2o f5 c3 oo
838o: oo 3e o1 32 f9 87 11 fc 87 cd od 82 cd 8c 84 db
839o: o1 e6 2o 2o 2b 2a fc 87 ed 5b fa 87 af ed 52 fa
83ao: ac 83 cd f6 83 7c c6 o7 67 c3 53 84 11 oo oo eb
83bo: af ed 52 cd f6 83 af 3e 64 9d 6f 26 o6 c3 53 84
83co: ed 5b fa 87 2a fc 87 19 cd f6 83 7d b7 28 oa 3e
83do: o6 94 67 3e 64 95 6f 18 7a 3e o7 94 67 18 74 11
83eo: e6 87 3e e5 12 1b 3e 9d 12 1b 3e 65 12 1b 3e 6o
83fo: 12 1b af 12 1b e9 cd 19 84 cd 8c 84 dd 22 f7 87
84oo: cd 42 84 cd 8c 84 cd 19 84 cd 8c 84 3a f7 87 67
841o: dd 22 f7 87 3a f7 87 6f c9 3a fe 87 57 7d 6c 26
842o: oo 1e oo o6 1o dd 21 oo oo 29 17 d2 2f 84 2c dd
843o: 29 dd 23 b7 ed 52 d2 3c 84 19 dd 2b 1o eb af 5c
844o: 57 c9 o6 64 21 oo oo cb 38 3o o1 19 c8 eb 29 eb
845o: 18 f5 c9 e5 cd 8c 84 6c af 67 cd a3 82 eb e1 d5
846o: af 67 cd a3 82 d1 63 cd 9o o2 21 e4 87 7e 23 77
847o: 21 e3 87 7e 23 77 3e 4o 2b 77 af 21 e6 87 77 cd
848o: bd oo cd 97 oo 7b fe o5 cc eo 82 c9 c5 d5 e5 cd
849o: bd oo e1 d1 c1 c9 oo oo oo oo oo oo oo oo oo oo
```

Bild 3. Die zwei Programme zur Kalibrierung und zur eigentlichen pH-Messung. Die Software zur Spannungsmessung muß sich ebenfalls im Speicher befinden

```
7eoo:  cd c9 o1 21 54 7f 11 oo 3c o1 oe oo ed bo cd 49
7e1o:  oo fe 31 28 2c fe 32 28 o2 18 f3 cd c9 o3 11 oo
7e2o:  3c o1 1a oo ed bo 3e 93 d3 9f cd 49 oo fe 4a 2o
7e3o:  f9 cd 81 7e 1a cd fd 7e 13 2b 7c b5 ca 66 oo 18
7e4o:  f3 3e 9b d3 9f 21 95 7f 11 oo 3c o1 2d oo ed bo
7e5o:  cd 49 oo fe 4a 2o f9 cd 81 7e d5 e5 21 c2 7f 11
7e6o:  4o 3c o1 oc oo ed bo e1 d1 cd 3o 7f 12 13 2b 7c
7e7o:  b5 ca 66 oo d5 11 od oo cd 4e 7f d1 18 eb c3 66
7e8o:  oo cd c9 o1 23 7c 7f 11 oo 3c o1 oc oo ed bo cd
7e9o:  bo 7e cd c9 7e d5 cd c9 o1 21 88 7f 11 oo 3c o1
7eao:  od oo ed bo cd bo 7e cd c9 7e eb d1 af ed 52 c9
7ebo:  o6 o4 11 1o 3c d5 cd 49 oo d1 fe od c8 12 13 1o
7eco:  f4 cd 49 oo fe od c8 18 e7 21 1o 3c cd f2 7e cb
7edo:  27 cb 27 cb 27 cb 27 57 23 cd f2 7e b2 57 23 cd
7eeo:  f2 7e cb 27 cb 27 cb 27 cb 27 5f 23 cd f2 7e b3
7efo:  5f c9 7e fe 3a fa fa 7e d6 o7 d6 3o c9 f5 c5 e5
7foo:  cd o7 7f e1 c1 f1 c9 37 o6 o9 f5 d4 1c 7f dc 21
7f1o:  7f f1 1f 1o f5 cd 1c 7f cd 1c 7f c9 af d3 9e 18
7f2o:  o6 3e ff d3 9e 18 oo 21 11 oo 2b 7c b5 2o fb c9
7f3o:  d9 db 9e 17 3o fb o6 o8 11 o9 oo cd 4e 7f 11 14
7f4o:  oo cd 4e 7f db 9e 17 cb 19 1o f3 79 d9 c9 1b 7b
7f5o:  b2 2o fb c9 45 49 4e 28 31 29 2d 41 55 53 28 32
7f6o:  29 3f 45 4d 55 46 2o 41 55 46 2o 45 49 4e 47 41
7f7o:  42 45 21 2o 4f 4b 41 59 28 4a 29 3f 53 5o 45 49
7f8o:  43 48 45 52 41 4e 46 2e 53 5o 45 45 49 43 48 45
7f9o:  52 45 4e 44 45 45 4d 55 46 2o 41 55 46 2o 41 55
7fao:  53 47 41 42 45 21 4e 4f 43 48 2o 4e 49 43 48 54
7fbo:  2o 53 54 41 52 54 45 4e 21 2o 4f 4b 41 4b 28 4a
7fco:  29 3f 45 4d 55 46 2o 53 54 41 52 54 45 4e 21 oo
```

Bild 4. Ein Programm zum Datenaustausch zwischen EMUF und TRS-80

Bild 4 zeigt ein Maschinenprogramm, geschrieben für den TRS-80 Model I, das den Datenaustausch mit dem EMUF abwickelt. Die Aus- und Eingabe erfolgt über Bit 7 des Port B am 8255 (Adresse 9EH des TRS-80). Das kleine Programm ist dialogorientiert und erklärt sich von selbst. Es beginnt ab 32 256 (dezimal). Es wurden lediglich drei Routinen des TRS-80 verwendet, nämlich Zeicheneingabe (0049H), Zeichenausgabe (0033H) und Löschen des Bildschirms (01C9H). Durch Ändern dieser Routinen und des Bildschirmspeichers (3C00H), kann deshalb dieses Maschinenprogramm auf andere Z80-Systeme angepaßt werden. Zu beachten ist noch, daß die Datenübertragung ohne irgendwelche Handshakesignale arbeitet und deshalb die Zeitschleifen (im Listing unterstrichen) beim Einsatz eines anderen Computers ebenfalls angepaßt werden müssen.

Literatur

[1] *Götz, D.*: Z80-EMUF mit Display und Tastatur. mc 1984, Heft 9.

[2] Datenblatt ICL-7109, Intersil.

Thomas Schlenger-Klink

Der Basic-EMUF

Intels 8052AH-Basic-Prozessor ist für Steuerzwecke ideal

Bei vielen kommerziellen Anwendungen kommen heute Ein-Chip-Mikrocomputer zum Einsatz. Die Verwendung eines solchen Prozessors scheitert im Low-Cost-Bereich jedoch häufig am Fehlen eines geeigneten Entwicklungssystems. Das Herz des hier vorgestellten Basic-EMUF (Einplatinen-Mikrocomputer für universelle Festprogrammanwendungen) ist ein in Basic programmierbarer Ein-Chip-Prozessor, der für die unterschiedlichsten Einsatzbereiche geeignet ist. Zur Programmierung reicht bereits ein einfaches Terminal aus.

Die Reihe der bereits in mc vorgestellen EMUFs zeigt, daß es eine ganze Menge von Anwendungen für kleine Einplatinen-Computer gibt. Zum Einsatz dieser Spezial-Computer wird aber ein geeignetes Entwicklungssystem benötigt. Die Programmierung erfolgt im allgemeinen in Assembler und ist daher oftmals mit erheblichem Aufwand verbunden. Die Programm-Entwicklungszeit läßt sich bei Verwendung von Hochsprachen drastisch verkürzen. Geeignete Hochsprachen-Compiler stehen allerdings nur den wenigsten Anwendern zur Verfügung. Durch die Verwendung eines Basic-Interpreters ist es nahezu jedem Computer- Interessierten möglich, Software für den Basic-EMUF zu schreiben.

Entwicklung von Hochsprachen für Ein-Chip-Prozessoren

Der Wunsch nach einer einfachen Softwareentwicklung führte bei einigen Halbleiterherstellern zur Entwicklung von Mikroprozessoren, die in einer Hochsprache programmiert werden können. Von der Firma Zilog wird der Baustein Z8671 hergestellt, der einen einfachen Basic-Interpreter enthält. Eine Platine mit diesem Baustein wurde in mc vorgestellt [1]. Das interne ROM mit 2 KByte bietet allerdings nur wenigen einfachen Funktionen Platz. Die Firma Rockwell stellte mit den Bausteinen 65F11/12 sehr leistungsfähige Einchip-Forth-Prozessoren zur Verfügung. Zur Programmentwicklung war allerdings ein sogenanntes „Development-ROM“ erforderlich, das durch seinen recht hohen Preis bei Einzelanwendungen kaum in Frage kommt.

Die Firma Intel hat mit dem Baustein 8052AH-Basic einen Single-Chip-Prozessor herausgebracht, der durch seine außergewöhnlichen Eigenschaften nahezu ideal für kleine Steuerungsanwendungen ist. Dieser Baustein aus der 8051-Familie enthält einen umfangreichen Basic-Interpreter, der zum Beispiel auch mit Fließkommazahlen umgehen kann. Außerdem sind viele weitere Funktionen enthalten, die die Programmierung von Steuerungen vereinfachen. Mit dem Baustein ist es möglich, fast vollständig ohne Entwicklungssystem auszukommen. Zur Programmierung reicht ein einfaches Terminal; EPROMs können direkt per Basic-Befehl in der Anwenderschaltung gebrannt werden.

Der Baustein wurde 1985 in mc vorgestellt [2]. Heute gibt es die noch erweiterte Version 1.1 des Chips. Die Erweiterungen betreffen hauptsächlich das Verhalten des ICs nach einem Reset und das Software-Interface zu Assembler-Programmen. *Bild 1* zeigt eine Auflistung der vorhandenen Befehle. Der Baustein 8052AH-Basic enthält: 8 KByte ROM (Basic-Interpreter), 256 Byte RAM, serielle Schnittstelle, 3 Timer, Interruptlogik und ein 8-Bit-I/O-Port.

Erweiterungen des MCS-Basic-52, Version 1.1

Mit X-ON (Control Q) und X-OFF (Control S) ist es jetzt möglich, die Ausgabe bei LIST und PRINT anzuhalten. In der Version 1.0 konnte die Ausgabe nur abgebrochen werden, was bei längeren Listings sehr störte. Die Statements IDLE (warten auf Interrupt), LD@, ST@ (Laden und Abspeichern von Floating-Point-Zahlen an einer bestimmten Adresse), PGM (EPROM- Programmierung) und RROM (Ablauf eines Basic-Programms im EPROM) sind neu hinzugekommen. Weiterhin die Befehle (F)PROG3...(F)PROG6. Mit diesen Befehlen kann unter anderem das Löschen des RAM-Speichers nach RESET verhin-

```
RUN             BAUD            +
CONT            CALL            /
LIST            CLEAR           **
LIST#           CLEARS          *
LIST@ (V1.1)    CLEARI          -
NEW             CLOCK1          .AND.
NULL            CLOCK0          .OR.
RAM             DATA            .XOR.
ROM             READ            ABS()
XFER            RESTORE         INT()
PROG            DIM             SGN()
PROG1           DO-WHILE        SQR()
PROG2           DO-UNTIL        RND
PROG3 (V1.1)    END             LOG()
PROG4 (V1.1)    FOR-TO-STEP     EXP()
PROG5 (V1.1)    NEXT            SIN()
PROG6 (V1.1)    GOSUB           COS()
FPROG1          RETURN          TAN()
FPROG2          GOTO            ATN()
FPROG3 (V1.1)   ON-GOTO         =,>,>=,<,<=,<>
FPROG4 (V1.1)   ON-GOSUB        ASC()
FPROG5 (V1.1)   IF-THEN-ELSE    CHR()
FPROG6 (V1.1)   INPUT           CBY()
                LET             DBY()
                ONERR           XBY()
                ONEX1           GET
                ONTIME          IE
                PRINT           IP
                PRINT#          PORT1
                PRINT@ (V1.1)   PCON
                PH0.            RCAP2
                PH0.#           T2CON
                PH0.@ (V1.1)    TCON
                PH1.            TMOD
                PH1.#           TIME
                PH1.@ (V1.1)    TIMER0
                PGM (V1.1)      TIMER1
                PUSH            TIMER2
                POP             XTAL
                PWM             MTOP
                REM             LEN
                RETI            FREE
                STOP            PI
                STRING
                UI0
                UI1
                UO0
                UO1
                LD@ (V1.1)
                ST@ (V1.1)
                IDLE
                RROM
```

Bild 1. Die Kommandos (links), die Befehle (Mitte) und die Operatoren (rechts) des Basic-Chips

dert werden. Dadurch bleiben Programme und Daten in einem batteriegepufferten RAM auch nach RESET intakt. Mit Assemblerroutinen kann der Anwender den Basic-Befehlssatz durch selbst definierte Funktionen und Statements erweitern. Wie diese Erweiterungen durchgeführt werden, ist anhand vieler Beispiele in [3] gezeigt. Das Handbuch (200 Seiten) ist sehr ausführlich mit Beispielen belegt. Intel gibt darin alle Informationen über den internen Aufbau des Basic-Interpreters: Speicherbelegung, Datenspeicherung, EPROM-File-Format usw.

Die Basic-EMUF-Platine

Die Basic-EMUF-Platine wurde in erster Linie für Stand-Alone-Steuerungsanwendungen entwickelt. Das heißt, ein Gerät arbeitet, ohne daß der Benutzer erst ein Programm laden und starten muß. Auf der Platine wurden viele Interface-Baugruppen untergebracht. Trotzdem war es möglich ein sehr universelles Konzept zu entwerfen, das auch eigenen Ideen genug Spielraum gibt. Die Leistungsfähigkeit dieses kleinen Einplatinen-Computers entspricht ungefähr der eines frühen Homecomputers. Als Anwendungsbeispiele für den Basic-EMUF seien genannt: Heizungs- und Temperaturregler, Zeitsteuerungen, Meßdatenerfassung, Schnittstellenwandler (z. B. V.24/Centronics), Schrittmotorensteuerung, Amateurfunkanwendungen, Musikinstrumente, Modelleisenbahnsteuerungen usw.

Bild 2 zeigt das Blockschaltbild des Basic-EMUFs. Die Karte besteht aus zwei Teilen: CPU- und Peripherieteil. Das Layout der Platine wurde so gestaltet, daß beide Teile mechanisch voneinander getrennt werden können. Das CPU-Modul kann als Baugruppe einzeln verwendet werden, wenn der Peripherieteil nicht den gewünschten Erfordernissen entspricht. Folgende Funktionen sind auf der vorliegenden Karte (Bausatz beim Autor erhältlich) enthalten:

CPU-Modul
- Zentraleinheit 8052AH-Basic
- 8...32 KByte RAM
- max. 16 KByte EPROM für Assemblerprogramme
- max. 16 KByte EPROM für Basicprogramme
- Adreßdecoder-PAL
- EPROM-Programmer
- V.24-Schnittstelle Terminal
- V.24-Schnittstelle Drucker
- 24-Bit-TTL-Interface (8255)
- RESET-Erzeugung
- Schaltung für Batteriepufferung
- I/O-Pfostensteckverbinder

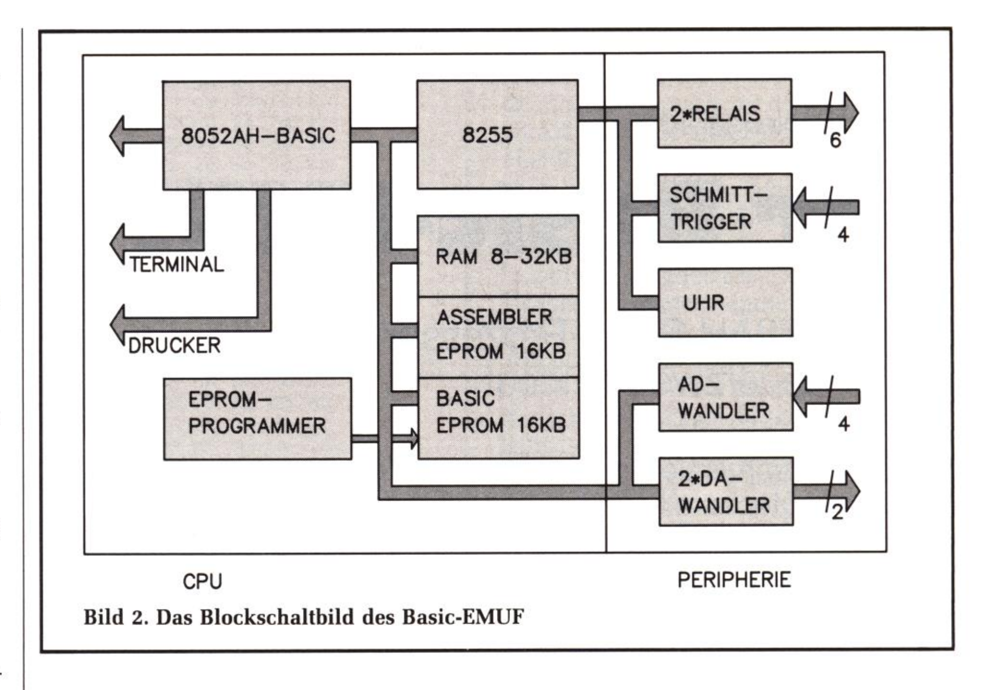


Bild 2. Das Blockschaltbild des Basic-EMUF

Peripherie-Modul 1
- 2 Relaisausgänge
- 4 entprellte CMOS-Eingänge
- 4-Kanal-12-Bit-AD-Wandler
- 2-Kanal-8-Bit-DA-Wandler
- Echtzeituhr
- Akku für RAM und Uhr
- kleines Lochrasterfeld

Die Schaltung des CPU-Moduls

Bild 3 zeigt die Schaltung des CPU-Moduls. Zur Stromversorgung dieses Teils genügt bereits eine Spannung von 5 V. Die Stromaufnahme beträgt etwa 250 mA (hängt von den eingesetzten Speicherbausteinen ab). Die CPU verfügt über einen gemultiplexten Daten- und Adreßbus. Das IC 74LS373 trennt Daten (D0-7) und Adressen (A0-7). Standardperipherie und -Speicherbausteine können damit angeschlossen werden. Das Modul ist nicht nur für den Baustein 8052AH-Basic geeignet, sondern auch für die Chips 8031, 8032, 8X51, 8X52, 9761 usw. Zum Abschalten des internen ROMs muß der Anschluß 31 der CPU auf LOW-Pegel gelegt werden. Die drei vorgesehenen Speicherbausteine sind je nach Steckplatz für unterschiedliche Speicherbereiche und -typen ausgelegt. Bei Verwendung des Basic-Chips ist ein Steckplatz für RAM, einer für ein Basic-EPROM und einer für ein reines Assembler-EPROM vorgesehen. Das RAM kann sowohl mit 8-KByte- als auch mit 32-KByte-Bausteinen bestückt werden. Da zu erwarten ist, daß die Preise für statische RAMs mit 32 KByte noch stark fallen, ist ein einziger RAM-Steckplatz wohl in allen Fällen ausreichend. Der Jumper J4 muß bei Einsatz von 32 KByte RAM kurzgeschlossen werden. Er befindet sich auf der Lötseite der Platine. Die Verwendung des Assembler-EPROMs ist erforderlich, wenn eigene Befehlssatzerweiterungen programmiert werden sollen. Das Basic-EPROM ist zusätzlich mit einer Schaltung zur EPROM-Programmierung „On Board" ausgerüstet. Das können weitere Assembler-Routinen sein. Zur Adreßdecodierung wird ein PAL vom Typ 10L8 eingesetzt. Die Decodierung kann damit in weiten Grenzen an eigene Erfordernisse angepaßt werden. *Bild 4* zeigt die logischen Gleichungen für zwei PALs, Version 1.1 zum Einsatz von 8 KByte RAM, Version 1.2 für 32 KByte RAM. Die Signale /ADDA, /RAMST und /PIOST sind zusätzliche Selectleitungen, die auf die beiden Pfostenverbinder für Erweiterungen (ST4 und ST5) geführt sind. Das IC MAX232 von Maxim ist ein kompletter Treiberbaustein für die V.24-Schnittstelle. Er enthält je zwei V.24-Sender und -Empfänger. Die positive und negative Spannung für die Sender werden auf dem Chip selbst erzeugt. Der Baustein dient zum Anschluß eines Terminals für die Programmentwicklung oder zur Anzeige von Daten. Ein serieller Drucker kann ebenfalls angeschlossen werden. In der Schaltung wurde ein zusätzlicher Eingang vorgesehen, um den Drucker auf seine Empfangsbereitschaft zu testen. Diese Funktion wird jedoch nicht vom internen Basic-Interpreter unterstützt, kann aber vom Anwender programmiert werden. Mit den Jumpern J1 und J2 kann der Pegel der Druckerschnittstelle gewählt werden. Diese

Möglichkeit wurde vorgesehen, um ein Miniatur-Epson-Druckwerk direkt anschließen zu können. Diese Druckwerke haben häufig ein serielles TTL-Interface. Die parallele Schnittstelle mit dem Baustein 8255 stellt dem Anwender 24 TTL-Leitungen zur freien Verfügung.

Die RESET-Erzeugung mit dem Baustein TL7705 hat gegenüber einem einfachen RC-Glied den Vorteil, daß immer ein RESET-Signal erzeugt wird, sobald die Betriebsspannung unter 4,6 V sinkt. Dadurch ist zum Beispiel sichergestellt, daß die CPU beim Ein- bzw. Ausschalten im rückgesetzten Zustand bleibt, um das versehentliche Verändern von Daten in einem akkugepufferten RAM zu verhindern.

Die Schaltung des Peripherie-Moduls

In *Bild 5* ist die Schaltung des Peripheriemoduls auf der Platine dargestellt. Bei der Auswahl der Echtzeituhr RTC58321 war entscheidend, daß dieser Typ über einen eingebauten Taktoszillator ver-

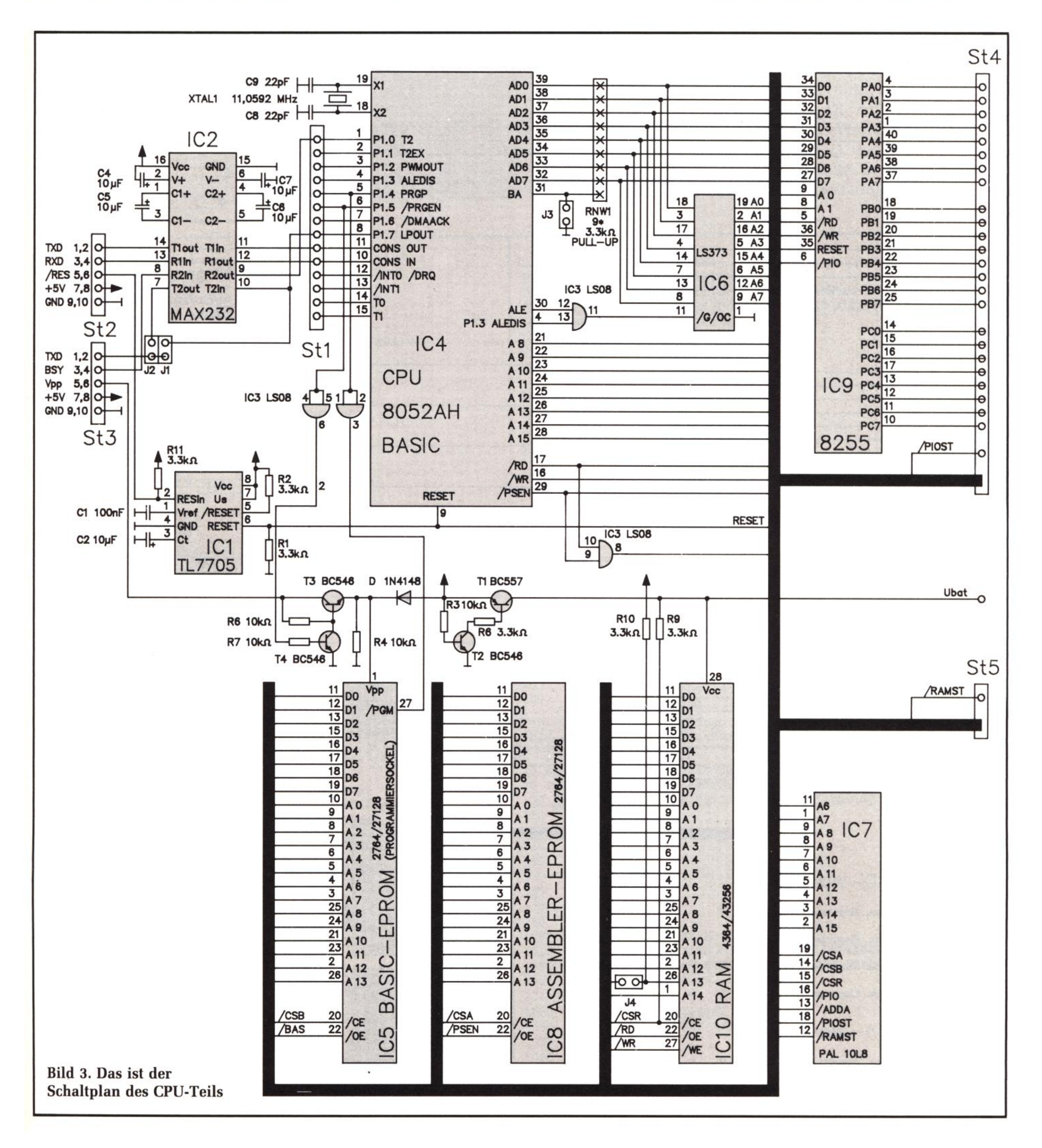


Bild 3. Das ist der Schaltplan des CPU-Teils

fügt, der Stromverbrauch im abgeschalteten Zustand typ. 6 µA beträgt und das Schreibsignal hardwaremäßig gesperrt werden kann. Da diese Uhr jedoch eine Zugriffszeit von 2 µs besitzt, die einen Anschluß direkt am Datenbus des Prozessors nicht zuläßt, wurde ein Teil der PIO 8255 für den Anschluß verwendet (PC 0...6). Der Jumper J5 auf der Karte verhindert das Verstellen der Uhr. Die Uhr kann bei gezogenem Jumper (J5) auch nicht durch Programmfehler verstellt werden. Diese Eigenschaft ist besonders bei protokollierenden Steuerungen wichtig. Der AD-Wandler vom Typ µPD 7002 wurde schon bei Schaltungen in mc verwendet [4]. Die Eingänge des AD-Wandlers können bei Bedarf mit Spannungsteilern und Tiefpässen (siehe Stückliste) bestückt werden und so fast beliebige Eingangssignale verarbeiten. Die Geschwindigkeit von ca. 5 ms bei 12 Bit Auflösung ist bei der Verwendung des Basic-Interpreters mehr als ausreichend. Die unbenutzten Eingänge des AD-Wandlers sollten später im Betrieb gegen Masse kurzgeschlossen werden,

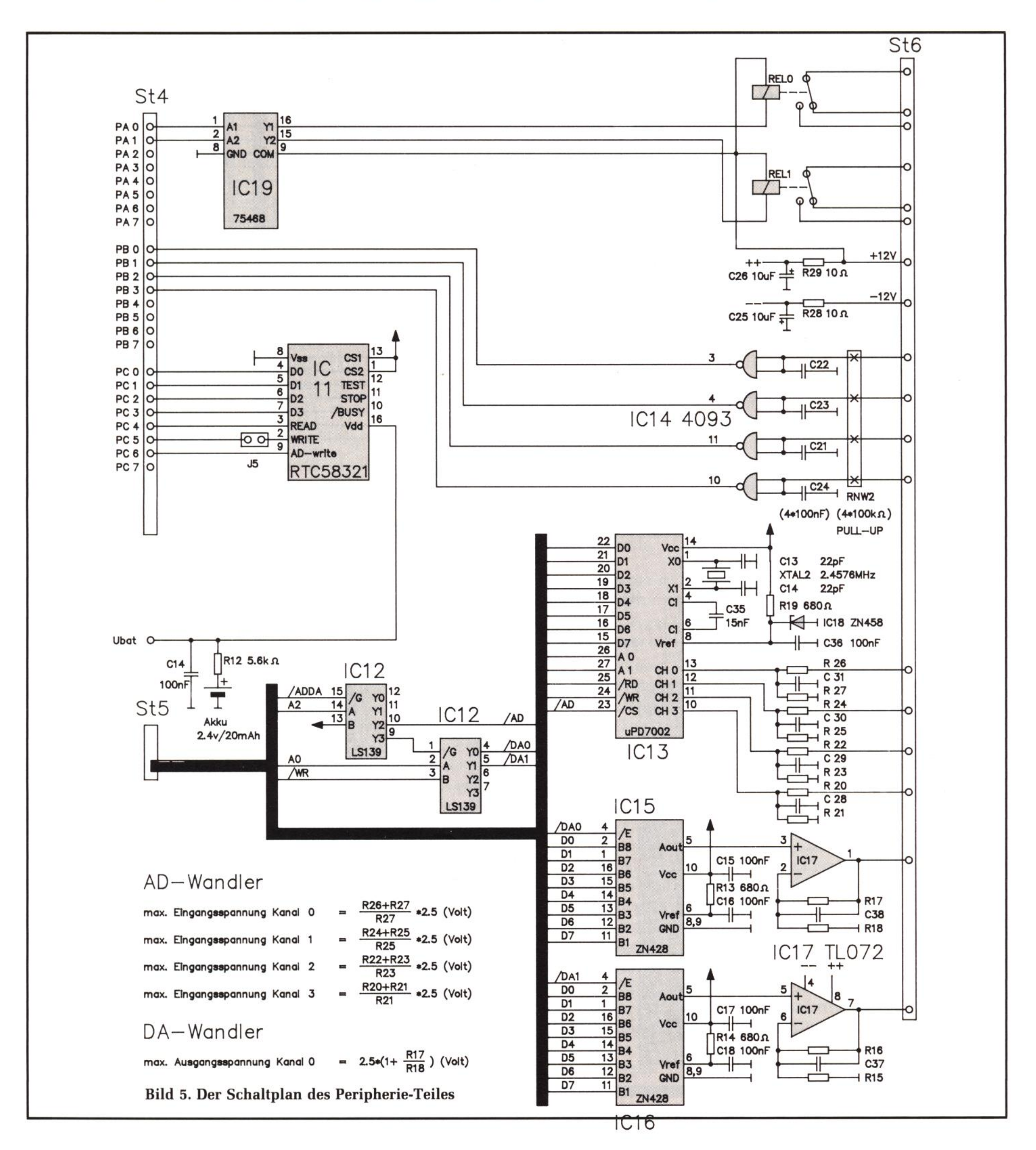


Bild 5. Der Schaltplan des Peripherie-Teiles

Tabelle 1: CPU-Signale an Stecker 1:

1	T2	Timer2 Trigger-Eingang (BUSY Eingang für Drucker)
2	T2EX	Timer2 externer Eingang
3	PWM	Ausgang für PWM Basic-Statement
4	/ALEDIS	Steuersignal zur EPROM-Programmierung
5	/Pr.Pulse	Programmierimpuls
6	/Pr.Ena.	Freigabe der Programmierspannung
7	P1.6	
	L Port 1.6 der CPU	
8	LPTO	Sendedaten zum Drucker TTL
9	RESET	RESET Signal (aktiv HIGH)
10	CONIN	Terminal Eingang TTL
11	CONOUT	Terminal Ausgang TTL
12	/INT0	Interrupt Eingang 0
13	/INT1	Interrupt Eingang 1
14	T0	Timer0 externer Eingang
15	T1	Timer1 externer Eingang

um eine Zerstörung des Wandlers zu vermeiden.
Die beiden DA-Wandler vom Typ ZN 428 zeichnen sich durch ihren einfachen Aufbau aus. Den beiden Wandlern ist zur Pufferung ein Operationsverstärker nachgeschaltet, der die Ausgangsspannung der DA-Wandler auf den gewünschten Wert verstärkt (0–10 V veränderbar, siehe Stückliste), und im Falle eines Falles die DA-Wandler vor einer Zerstörung schützt. Die negative Spannung ist für die DA-Wandler notwendig, wenn sie bis herunter auf 0 V betrieben werden sollen. Die Adreßdecodierung für AD- und DA-Wandler wird durch das Signal /ADDA und den Baustein 74LS139 bestimmt. Die entprellten CMOS-Eingänge sind mit dem 4fach-

```
PAL10L8
VERSION 1.1 8kB RAM
8052 BASIC DEKODER-PAL 29.7.86

A7 A15 A14 A13 A12 A11 A10 A9 A8 GND
A6 /RAMST /ADDA /CSB /CSR /PIO NC /PIOST /CSA VCC

PIO=A15*A14*A13*/A11*/A10*/A9*/A8*/A7*/A6
RAMST=A15*A14*A13*/A11*/A10*/A9*/A8*/A7*A6
PIOST=A15*A14*A13*/A11*/A10*/A9*/A8*A7*/A6
ADDA=A15*A14*A13*/A11*/A10*/A9*/A8*A7*A6
CSR=/A15*/A14*/A13
CSB=A15*/A14
CSA=/A15


DESCRIPTION:
PIO    CS 8255                                    E000-E03F
RAMST  CS AN RAM-STECKER                          E040-E07F
PIOST  CS AN PIO-STECKER                          E080-E0BF
ADDA   CS AD/DA WANDLER                           E0C0-E0FF
CSR    CS RAM (8K)                                0000-1FFF
CSA    CS ASSEMBLER EPROM (32K)                   0000-7FFF
CSB    CS BASIC EPROM (PROGRAMMIERSOCKEL 16K)     8000-BFFF


PAL10L8
VERSION 1.2 32kB RAM
8052 BASIC DEKODER-PAL 29.7.86

A7 A15 A14 A13 A12 A11 A10 A9 A8 GND
A6 /RAMST /ADDA /CSB /CSR /PIO NC /PIOST /CSA VCC

PIO=A15*A14*A13*/A11*/A10*/A9*/A8*/A7*/A6
RAMST=A15*A14*A13*/A11*/A10*/A9*/A8*/A7*A6
PIOST=A15*A14*A13*/A11*/A10*/A9*/A8*A7*/A6
ADDA=A15*A14*A13*/A11*/A10*/A9*/A8*A7*A6
CSR=/A15
CSB=A15*/A14
CSA=/A15


DESCRIPTION:
PIO    CS 8255                                    E000-E03F
RAMST  CS AN RAM-STECKER                          E040-E07F
PIOST  CS AN PIO-STECKER                          E080-E0BF
ADDA   CS AD/DA WANDLER                           E0C0-E0FF
CSR    CS RAM (32K)                               0000-7FFF
CSA    CS ASSEMBLER EPROM (32K)                   0000-7FFF
CSB    CS BASIC EPROM (PROGRAMMIERSOCKEL 16K)     8000-BFFF
```

Bild 4. Die PAL-gleichungen für das Decoder-PAL

```
10    REM PIO TESTPROGRAMM
20    A=0E000H :  REM BASISADRESSE PIO
30    PRINT "ERZEUGUNG EINER RECHTECKSPANNUNG AN PA,PB UND PC"
40    XBY(A+3)=80H :  REM INITIALISIERUNG ALLE PORTS ALS AUSGAENGE
100   XBY(A)=0 :  REM PORT A
110   XBY(A+1)=0 :  REM PORT B
120   XBY(A+2)=0 :  REM PORT C
130   GOSUB 1000 :  REM DELAY
140   XBY(A)=255
150   XBY(A+1)=255
160   XBY(A+2)=255
170   GOSUB 1000 :  REM VERZOEGERUNG
180   GOTO 100
1000   REM VERZOEGERUNG
1010   FOR I=0 TO 200 :  NEXT I
1020   RETURN
```

Bild 6. Der PIO-Test

```
10     PRINT "TESTPROGRAMM FUER CMOS-EINGAENGE"
20    A=0E000H :  REM BASISADRESSE PIO
30     XBY(A+3)=0FFH : REM PA,PB,PC EINGAENGE
40     PRINT "EINGANG 0123"
50     FOR I=0 TO 3
60    M=2**I : W=XBY(A+1).AND.M :  REM MASKIEREN EINES EINGANGS
70     IF W=0 THEN  PRINT "H", ELSE  PRINT "L",
80     NEXT I
90     PRINT  CR ,"         ",
100    GOTO 50
```

Bild 7: Test für die CMOS-Eingänge

```
10     REM ADC TESTPROGRAMM
20    A=0E0C0H :  REM BASISADRESSE AD-WANDLER
100    PRINT "LAUFENDE MESSUNG ALLER 4 KANAELE"
110    FOR I=0 TO 3
120    PUSH I
130    GOSUB 1000
140    POP W
150    PRINT "KANAL",I,"=",W," ",
160    NEXT I
170    PRINT  CR ,
180    GOTO 110
1000   REM MESSEN EINES AD-KANALS
1010   REM DER KANAL WIRD AUF DEM STACK UEBERGEBEN
1020   REM DER MESSWERT WIRD AUF DEM STACK ZURUECKGEGEBEN
1030   POP T1
1040  XBY(A)=T1.OR.8
1050   IF (XBY(A).AND.64)=0 THEN 1050
1060  T1=16*XBY(A+1)+XBY(A+2)/16
1070   PUSH T1
1080   RETURN
```

Bild 8. Der Test für den AD-Wandler

```
10     REM DAC TESTPROGRAMM
20    A=0E0C4H :  REM BASISADRESSE DA-WANDLER
100    PRINT "DA-WANDLER TESTPROGRAMM"
110    INPUT "KANALNUMMER (0/1) ? ",K
120    INPUT "AUSGABEWERT (0-255) ? ",W
130    XBY(A+K)=W
140    PRINT  :  GOTO 110
```

Bild 9. Ein DA-Wandler-Test

```
10     REM RTC TESTPROGRAMM
20     REM ACHTUNG! BEACHTE STUNDEN ZEHNER
30    A=0E000H :  REM BASISADRESSE PIO
40    XBY(A+3)=93H :  REM INITIALISIERUNG
100    REM STELLEN DER UHR MIT DEN DATEN IN ZEILE 5000
110    PRINT "ZUM STELLEN DER UHR JUMPER J5 EINSETZEN"
120    FOR I=12 TO 0 STEP -1
130    READ W :  PUSH W :  REM ZU SCHREIBENDE DATEN
140    PUSH I :  REM ADRESSE
150    GOSUB 1200
160    NEXT I
200    REM DATUM UND UHRZEIT
210    PRINT "DATUM    UHRZEIT"
220    GOSUB 2000
230    GOTO220
1000   REM ADRESS-STROBE
1010   REM ADRESSE AUF DEM STACK
1020  XBY(A+3)=92H :  REM UMSCHALTUNG PC0-3 AUSGANG
1030   POP A1 :  REM ADRESSE
1040  XBY(A+2)=A1 :  REM AUSGABE DER ADRESSE
1050  XBY(A+3)=0DH :  REM SETZEN ADRESS-STROBE
1060  XBY(A+3)=0CH :  REM RUECKSETZEN ADRESS-STROBE
1070   RETURN
1100   REM LESEN EINES UHREN-WERTES
1110   GOSUB 1000 :  REM ADRESS AUSGEBEN
1120  XBY(A+2)=0FH
1130  XBY(A+3)=93H :  REM UMSCHALTEN AUF EINGANG
1140  XBY(A+3)=9 :  REM LESEN EINSCHALTEN
1150  W1=XBY(A+2) : W1=W1.AND.0FH :  REM EINLESEN
1160  XBY(A+3)=8 :  REM LESEN AUSSCHALTEN
1170   PUSH W1 :  REM DATEN AUF DEN STACK
1180   RETURN
1200   REM SCHREIBEN EINES UHRENWERTES
1210   REM DATEN UND ADRESSE AUF DEM STACK
1220   GOSUB 1000 :  REM ADRESSE AUSGEBEN
1230   POP W1 : XBY(A+2)=W1 :  REM DATEN AUSGEBEN
1240  XBY(A+3)=0BH :  REM SCHREIBEN EINSCHALTEN
1250  XBY(A+3)=0AH :  REM SCHREIBEN AUSSCHALTEN
1260   RETURN
2000   REM AUSLESEN UND ANZEIGEN VON DATUM UND UHRZEIT
2010   PUSH 7 :  GOSUB 3000 :  REM TAG
2020   PRINT ".", :  PUSH 9 :  GOSUB 3000 :  REM MONAT
2030   PRINT ".", :  PUSH 11 :  GOSUB 3000 :  REM JAHR
2040   PRINT " ", :  PUSH 5 :  GOSUB 1100
2050   POP W1 : W1=W1.AND.3 : W1=W1.OR.30H
2060   PRINT CHR(W1), :  REM STUNDEN ZEHNER
2070   PUSH 4 :  GOSUB 1100
2080   POP W1 : W1=W1.OR.30H :  PRINT CHR(W1), :  REM STUNDEN EINER
2090   PRINT ":", :  PUSH 2 :  GOSUB 3000 :  REM MINUTEN
2100   PRINT ":", :  PUSH 0 :  GOSUB 3000 :  REM SEKUNDEN
2110   PRINT  CR ,
2120   RETURN
3000   REM AUSGABE VON 2 ZEICHEN DER RTC
3010   POP A1 :  PUSH A1 :  GOSUB 1100
3020  A1=A1+1 :  PUSH A1 :  GOSUB 1100
3030   POP W1 : W1=W1.OR.30H :  PRINT CHR(W1),
3040   POP W1 : W1=W1.OR.30H :  PRINT CHR(W1),
3050   RETURN
5000   DATA 8,6 : REM JAHR
5010   DATA 0,8 : REM MONAT
5020   DATA 0,1 : REM TAG
5030   DATA 5 : REM WOCHENTAG
5040   DATA 9,9 : REM STUNDEN (10*STD+8 BEI 24 STD FORMAT)
5050   DATA 3,0 : REM MINUTEN
5060   DATA 0,0 : REM SEKUNDEN
```

Bild 10. Das Testprogramm für die Echtzeit-Uhr

Tabelle 2: V.24-Schnittstellen

V.24 an Stecker 2: Terminal		
1	2	Sendedaten Termial
3	4	Empfangsdaten Terminal
5	6	Reset-Eingang (aktiv LOW)
7	8	+ 5 V
9	10	GND
V.24 an Stecker 3: Drucker		
1	2	Sendedaten Drucker
3	4	BUSY vom Drucker (wird nicht von MCS Basic 52 bedient)
5	6	Programmierspannung für EPROM
7	8	+ 5 V
9	10	GND

Tabelle 3: PIO an Stecker 4

1	PA 4	2	PA 3
3	PA 5	4	PA 2
5	PA 6	6	PA 1
7	PA 7	8	PA 0
9	/WR	10	/RD
11	RESET	12	/PIOST
13	D 0	14	GND
15	D 1	16	A1
17	D 2	18	A0
19	D 3	20	PC 7
21	D 4	22	PC 6
23	D 5	24	PC 5
25	D 6	26	PC 4
27	D 7	28	PC 3
29	+5V	30	PC 2
31	PB 7	32	PC 1
33	PB 6	34	PC 0
35	PB 5	36	PB 0
37	PB 4	38	PB 1
39	PB 3	40	PB 2

Tabelle 4: RAM-Stecker ST5

1	+Ubat	2	+Ubat
3	+5V	4	+5V
5	GND	6	GND
7	/WR	8	/CS ADDA
9		10	A 5
11		12	A 4
13	/RD	14	A 3
15	RESET	16	A 2
17	/RAMST	18	A 1
19	D 7	20	A 0
21	D 6	22	D 0
23	D 5	24	D 1
25	D 4	26	D 2
27	D 3	28	GND

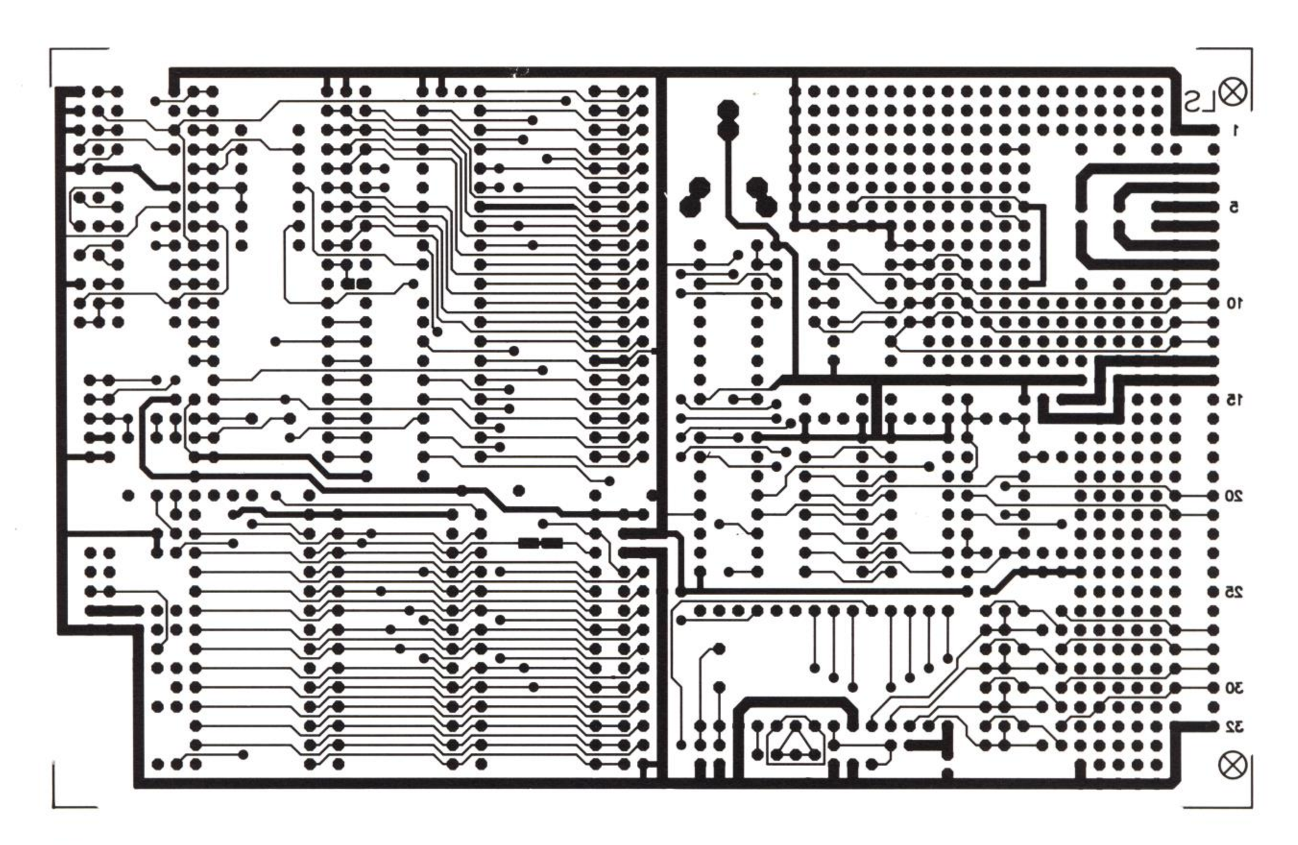


Bild 11. Die Bestückungsseite der Platine (oben), die Lötseite (unten)

NAND-Schmitt-Trigger 4093 aufgebaut. Zu entprellende Kontakte werden einfach zwischen Eingang und Masse geschaltet. Beim ersten Schließen des Kontaktes wird der Eingangskondensator sehr schnell entladen. Die Aufladung erfolgt über einen Pullup-Widerstand.

Dies hat zur Folge, daß bei einer geeignete Wahl der Bauteile die Spannung beim Prellen des Kontaktes nicht die Schaltschwelle des Schmitt-Triggers erreichen kann. Bei mechanischen Kontakten (Schalter, Taster, Relais) hat sich die Kombination 100 kΩ mit 100 nF bestens bewährt. Die beiden Relais sind mit integrierten Transistor-Schaltern (75468) an die PIO 8255 (PA0,PA1) angeschlossen. Die restlichen Treiber dieses Bausteins können bei Bedarf für eigene Anwendungen verwendet werden.

Die Anschlüsse von AD-, DA-Wandler, CMOS-Eingängen, Relaisausgängen und Stromversorgung sind auf eine 64polige VG-Leiste (ST5) geführt.

Aufbau des Basic-EMUF

Beim Aufbau der Platine sind die üblichen Regeln für den Aufbau von elektronischen Schaltungen zu beachten. Um die Bauhöhe der Karte möglichst niedrig

Tabelle 5: VG-Leiste 64polig (ST6)

a	Nr.	c
+5 V	1	+5 V
–	2	–
Relais 1 gem. Ansch.	3	Relais 1 gem. Anschluß
Relais 1 Öffner	4	Relais 1 Öffner
Relais 1 Schließer	5	Relais 1 Schließer
Relais 0 Schließer	6	Relais 0 Schließer
Relais 0 Öffner	7	Relais 0 Öffner
Relais 0 gem. Ansch.	8	Relais 0 gem. Anschluß
CMOS Eingang 0	9	CMOS Eingang 0
CMOS Eingang 1	10	CMOS Eingang 1
CMOS Eingang 2	11	CMOS Eingang 2
CMOS Eingang 3	12	CMOS Eingang 3
+12 V	13	+12 V
−12 V	14	−12 V
–	15	–
–	16	–
–	17	–
–	18	–
–	19	–
DA-Ausgang 0	20	DA-Ausgang 0
DA-Ausgang 1	21	DA-Ausgang 1
–	22	–
–	23	–
–	24	–
–	25	–
–	26	–
AD-Eingang 0	27	AD-Eingang 0
AD-Eingang 1	28	AD-Eingang 1
AD-Eingang 2	29	AD-Eingang 2
AD-Eingang 3	30	AD-Eingang 3
	31	
0 V	32	0 V

– = Zur Zeit nicht belegt, frei für eigene Anwendungen

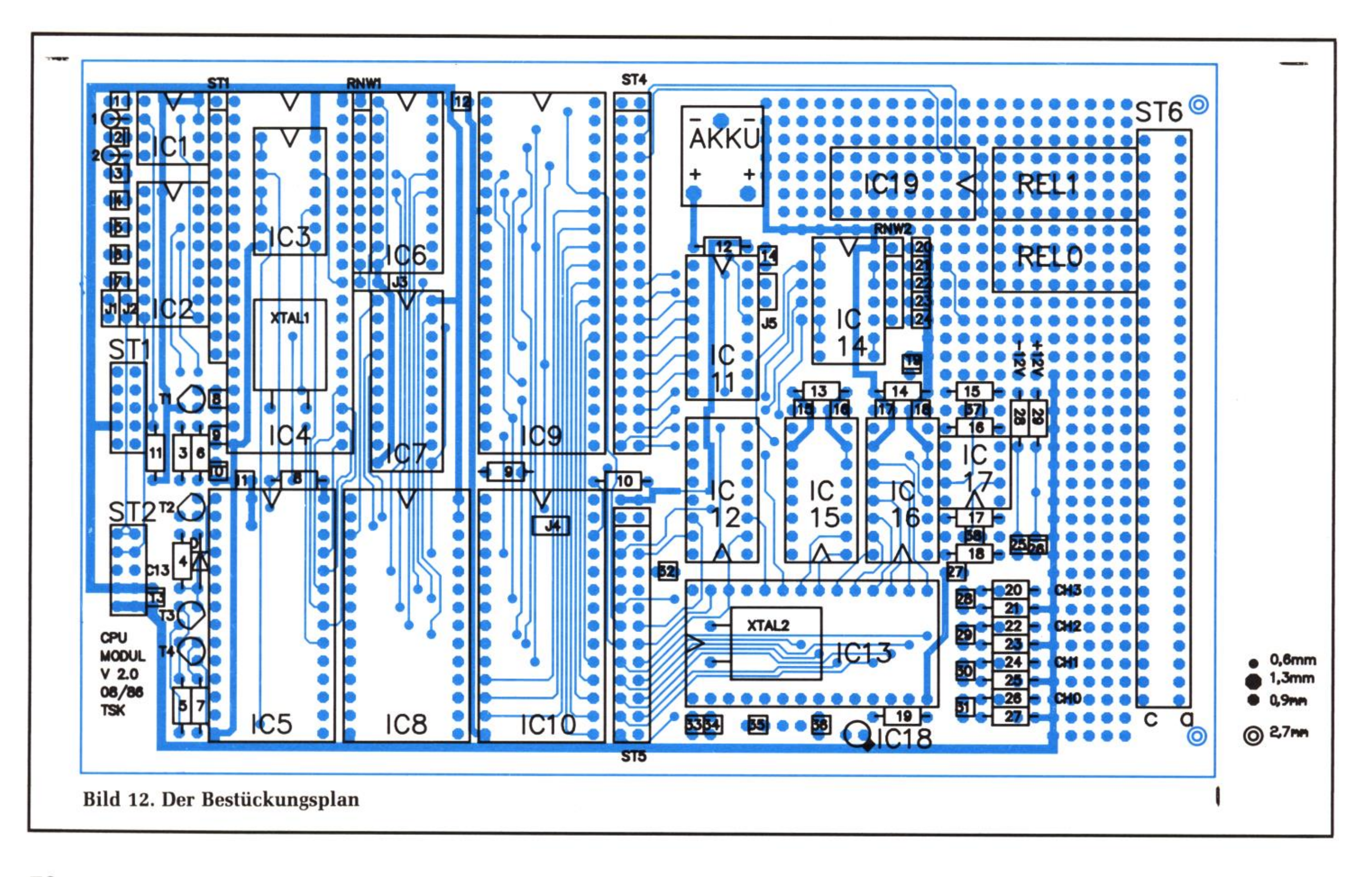


Bild 12. Der Bestückungsplan

Tabelle 6: Jumperbelegung

J1	Drucker-Ausgang TTL-Pegel
J2	Drucker-Ausgang V.24-Pegel
J3	Sperren des internen ROM
J4	Lötbrücke für 32 KB RAM (auf der Lötseite)
J5	Schreibfreigabe für RTC 58321

zu halten, wurden die Quarze für CPU und AD-Wandler unter die zugehörigen Sockel gelegt. Es sollten dafür nur hochwertige IC-Sockel verwendet werden. Bei einigen Typen sind die Mittelstege herauszutrennen. Die Quarze werden jeweils mit einem kleinen Stück doppelbeschichteten Klebeband befestigt. Ein weiterer Vorteil dieser Bestückung ist, daß die Quarze sehr gut gegen mechanische Beschädigungen geschützt sind. Das IC 74LS08 ist ebenfalls unter dem CPU-Sockel untergebracht. Dieser Baustein sollte direkt in die Platine eingelötet werden. Nach der Bestückung der Bauteile des CPU-Teils (Stückliste in *Tabelle 7*) muß die Platine sehr sorgfältig auf Lötfehler untersucht werden. Der Akku sollte erst ganz zum Schluß eingesetzt werden, wenn die Funktion der Karte überprüft worden ist. Die Bestükkung erfolgt am besten für CPU- und Peripherieteil getrennt. Es vereinfacht sich dadurch der Funktionstest der Schaltung.

Tabelle 7: Stückliste CPU-Modul

C1,3,11,12	100 nF Vielschicht
C2,4,5,6,7,10	10 µF, 16 V, Tantal
C8,9	22 pF Kerko
R1,2,6,8,9,10	3,3 kΩ
R3,4,5,7	10 kΩ
RNW1	9 mal 3,3 kΩ, SIL-Netzwerk
IC 1	TL 7705
IC 2	MAX232
IC 3	74LS08 (auch HCT)
IC 4	8052AH-Basic
IC 5	Basic-EPROM (Programmiersockel)
IC 6	74LS373 (auch HCT)
IC 7	PAL 10L8
IC 8	Assembler-EPROM
IC 9	8255
IC 10	RAM 4364, 43256 o.ä.
T1	BC 557 o.ä., PNP
T2,3,4	BC 546 o.ä. NPN
D	1N4148
XTAL1	11,0592 MHz, HC-18U
ST1	15pol. Pfostenstecker, 1reihig
ST 2,3	10pol. Pfostenstecker, 2reihig
ST4	40pol. Pfostenstecker, 2reihig
ST5	26pol. Pfostenstecker, 2reihig

Die Inbetriebnahme des Basic-EMUFs

Die Stecker- und Jumperbelegungen sind den *Tabellen 1* bis *6* zu entnehmen. Zur Inbetriebnahme sollte mindestens ein Vielfachmeßgerät, besser noch ein Oszilloskop zur Verfügung stehen. Weiterhin ist eine einstellbare Stromversorgung (4 V...5,5 V, max.0,5 A) von Vorteil. Nach nochmaliger Untersuchung der Platine auf evtl. Lötfehler wird zunächst die einstellbare 5-V-Versorgung (4 V bis 5 V) angeschlossen. Die Stromaufnahme darf nur wenige mA betragen. Nach dem Einsetzen des TL7705 kann am Anschluß 9 der CPU das RESET-Signal beobachtet werden. Dieses sollte bei Erhöhung der Versorgungsspannung, angefangen bei 4 V, bei etwa 4,6 V vom HIGH- in den LOW-Zustand wechseln. Ist dieser Test erfolgreich verlaufen, wird die CPU 8052AH-Basic bestückt. Mit einem Oszilloscop kann an den Anschlüssen 18 und 19 die Funktion des Taktoszillators nachgeprüft werden. Am Anschluß 11 des Sockels für den 74LS373 muß ein periodisches Signal mit dem Oszilloskop meßbar sein. Danach werden der 74LS373 und das RAM eingesetzt. In den Sockel des PALs ist eine Drahtbrücke von Anschluß 4 zu Anschluß 15 einzustecken. Bei ordnungsgemäßem Aufbau sollte jetzt nach dem Einschalten an einem angeschlossenen Terminal nach Betätigung der Space-Taste die Meldung:

```
*MCS-51(tm) BASIC V1.1*
READY >
```

erscheinen. Jetzt kann das PAL eingesetzt werden. Die CPU prüft beim Einschalten die Größe des RAM-Bereiches. Mit

```
>PRINT MTOP
```

Tabelle 8: Stückliste Peripherie-Modul

C14,15,16,17,18,20,27,32,36	100 nF, Vielschicht
C21,22,23,24	CMOS-Eingänge (100 nF)
C19,25,26	10 µF, 16 V, Tantal
C33,34	22 pF, keramisch
C35	15 nF, Folie
C28,29,30,31	nach Bedarf, AD-Wandler Eingangs-TP (100 nF)
C37,38	nach Bedarf, DA-Wandler Ausgangs-TP (10 nF)
R13,14,19	680 Ω
R15,18	10 kΩ (DA-Wandler-Verstärkung)
R16,17	20 kΩ (DA-Wandler-Verstärkung)
R20,22,24,26	Spannungsteiler AD-Wandler
R21,23,25,27	Spannungsteiler AD-Wandler
R28,29	10 Ω
RNW2	400 kΩ SIL-Netzwerk (CMOS-Eingänge)
IC 11	RTC 58321
IC 12	74LS139 (auch HCT)
IC 13	µPD 7002 C
IC 14	4093
IC 15,16	ZN 428
IC 17	TL072
IC 18	ZN 458
IC 19	75468
Akku	Varta 2,4 V, 20 mAh
REL0,1	Siemens V23102-A6-A111 o.ä.
XTAL2	2,4576 MHz, HC-18U
ST6	VG-Leiste, 64pol., a+c

Tabelle 9: Die Belegung des Bausteines 8255

PA 0	Relais 0
PA 1	Relais 1
PA 2	
PA 3	
PA 4	
PA 5	
PA 6	
PA 7	
PB 0	entprellter CMOS-Eingang 0
PB 1	entprellter CMOS-Eingang 1
PB 2	entprellter CMOS-Eingang 2
PB 3	entprellter CMOS-Eingang 3
PB 4	
PB 5	
PB 6	
PB 7	
PC 0	RTC Daten D0
PC 1	RTC Daten D1
PC 2	RTC Daten D2
PC 3	RTC Daten D3
PC 4	RTC Lesefreigabe RE
PC 5	RTC Schreibfreigabe WE
PC 6	RTC Adress-Strobe AWR

kann dieser Wert angezeigt werden. Er beträgt bei 8 KByte RAM 8191 Byte, bei 32 KByte RAM 32767 Byte. Die EPROM-Programmierlogik ist vor Einsatz eines EPROMs zu überprüfen. Bei eingeschaltetem Basic-EMUF wird die Programmierspannung von 12,5 bzw. 21 V angelegt, und die Spannung am Anschluß 1 (Vpp) des Programmiersockels gemessen. Sie sollte ca. 4,3 V betragen. Beim Kurzschließen des Anschlusses 6 der CPU gegen Masse steigt diese Spannung auf nahezu den Wert der angelegten Programmierspannung an (ca. 12,3 V bzw. 20,8 V). Nach Entfernen des Kurzschlusses wird die Spannung am Anschluß 27 (/PGM) des Programmiersockels getestet. Sie beträgt ca. 5 V.
Bei Kurzschluß des Anschlusses 5 der CPU gegen Masse fällt diese Spannung auf nahezu 0 V ab. Nach Entfernen der Kurzschlusses kann ein EPROM zum Test eingesetzt werden und ein kleines Programm in das EPROM programmiert werden. Die Funktion der PIO 8255 kann mit dem Programm in Bild 6 getestet werden. Erst wenn alle Teile des CPU-Moduls in Ordnung sind, werden die weiteren Peripheriebausteine eingesetzt (Stückliste, *Tabelle 8*). Die Testprogramme für die Peripherie aus *Bild 6* bis *10* können mit Hilfe des EPROM-Programmers in ein EPROM abgelegt werden. Dadurch wird die Neueingabe der Programme nach einem Reset überflüssig. Sie können mit dem Befehl „RROM“ gestartet werden.

Ausblick

Die universellen Einsatzmöglichkeiten des Basic-EMUF lassen erwarten, daß eine ganze Reihe von Hard- bzw. Softwareanwendungen entwickelt werden. Zur Zeit existiert bereits ein einfaches LC-Display (2 Zeilen zu je 16 Zeichen) mit Tastatur, das für die Ein- und Ausgabe in Steuerungen verwendet werden kann. Weiterhin ist eine I/O-Erweiterung mit 8255, Optoeingängen und Relaisausgängen vorhanden. Für die Programmentwicklung und zur Anzeige großer Datenmengen ist ein grafikfähiges Video-Display geplant, das direkt auf den Basic-EMUF aufgesteckt werden kann.

Literatur

[1] *Wiesemann, Reinhard*: Basic-Einplatinen-Compter. mc 1983, Heft 2.
[2] *Petersen, W.*: Basic an Board. mc 1985, Heft 8, S.83...85.
[3] Intel: MCS BASIC-52 USERS MANUAL. 1985.
[4] *Schön, Alfred*: 12-Bit-A/D-Wandler. mc 1983, Heft 11, S.100...102.
[5] Intel: Microcontroller Handbook,1986.
[6] NEC: Datenblatt µPD 7002.

Thomas Schlenger-Klink

Basic-EMUF mit LCD und Tasten

Bei vielen Anwendungen von Einplatinen-Computern ist die Ein-/Ausgabe von Daten während des Betriebs notwendig, so auch beim Basic-EMUF. Die Programmierung einer solchen Ein-Ausgabe-Einheit ist bei Realisierung in Maschinensprache relativ umfangreich und zeitaufwendig. Die hier vorgestellte Lösung LCDKEY ist einfach zu handhaben, da die Programmierung mit den Basic-Befehlen „PRINT" und „INPUT" erfolgt. Die Daten können dabei alphanumerisch angezeigt und eingegeben werden.

Die LCDKEY-Baugruppe wurde in erster Linie als Ein-/Ausgabe-Baugruppe für den Basic-EMUF entwickelt. Der Anschluß an andere Computer ist jedoch ebenfalls möglich. Die Ansteuerung der Tastatur und des LC-Moduls erfolgt über einen PIO-Baustein 8255. Dieser Baustein stellt 24 TTL Ein-Ausgabeleitungen zur Verfügung. Damit kann die Platine auch an andere Computer angeschlossen werden (z.B. 8085, Z80 etc.). Die Anzeige besteht aus einem fertig montierten LC-Modul, das in den verschiedensten Ausführungsformen erhältlich ist (mit und ohne Beleuchtung, 16-40 stellig). Die Taster des Tastenfeldes sind so konstruiert, daß sie mit einlegbaren Papierschildern sehr einfach und dauerhaft selbst beschriftet werden können, es sind jedoch auch fertig beschriftete Tastenkappen lieferbar.

Schaltungsbeschreibung

Bild 1 zeigt das Schaltbild des gesamten LCDKEY-Moduls. Das Anzeigemodul enthält bereits die gesamte Ansteuerungselektronik für das Display. Auf der Platine kann eine Anzeige mit zwei Zeilen zu je 16 Stellen montiert werden. Das Platinenformat wurde so gewählt, daß sich das LCDKEY-Modul sehr gut in einem 19-Zoll Einschubsystem unterbringen läßt. Andere Module (20, 32, 40 Stellen) werden über ein Kabel an die Grundplatine angeschlossen, da die unterschiedlichen Module meist die gleiche Ansteuerungselektronik enthalten. Das Anzeigemodul stellt den kompletten ASCII-Zeichensatz dar. Darüberhinaus verfügt die Anzeige über einen selbstprogrammierbaren Zeichengenerator zur Anzeige selbstentworfener Zeichen (z.B. Umlaute). Das LC-Modul ist über einen PIO-Baustein 8255 an den Datenbus des Basic-EMUF angeschlossen (über PIO-Stecker ST4). Dadurch werden auf dem Basic-EMUF keine I/O-Leitungen belegt. Der Datenbus der Anzeige ist an Port A des 8255 angeschlossen, die restlichen Steuerleitungen (RS, E, R/W) an Port C.

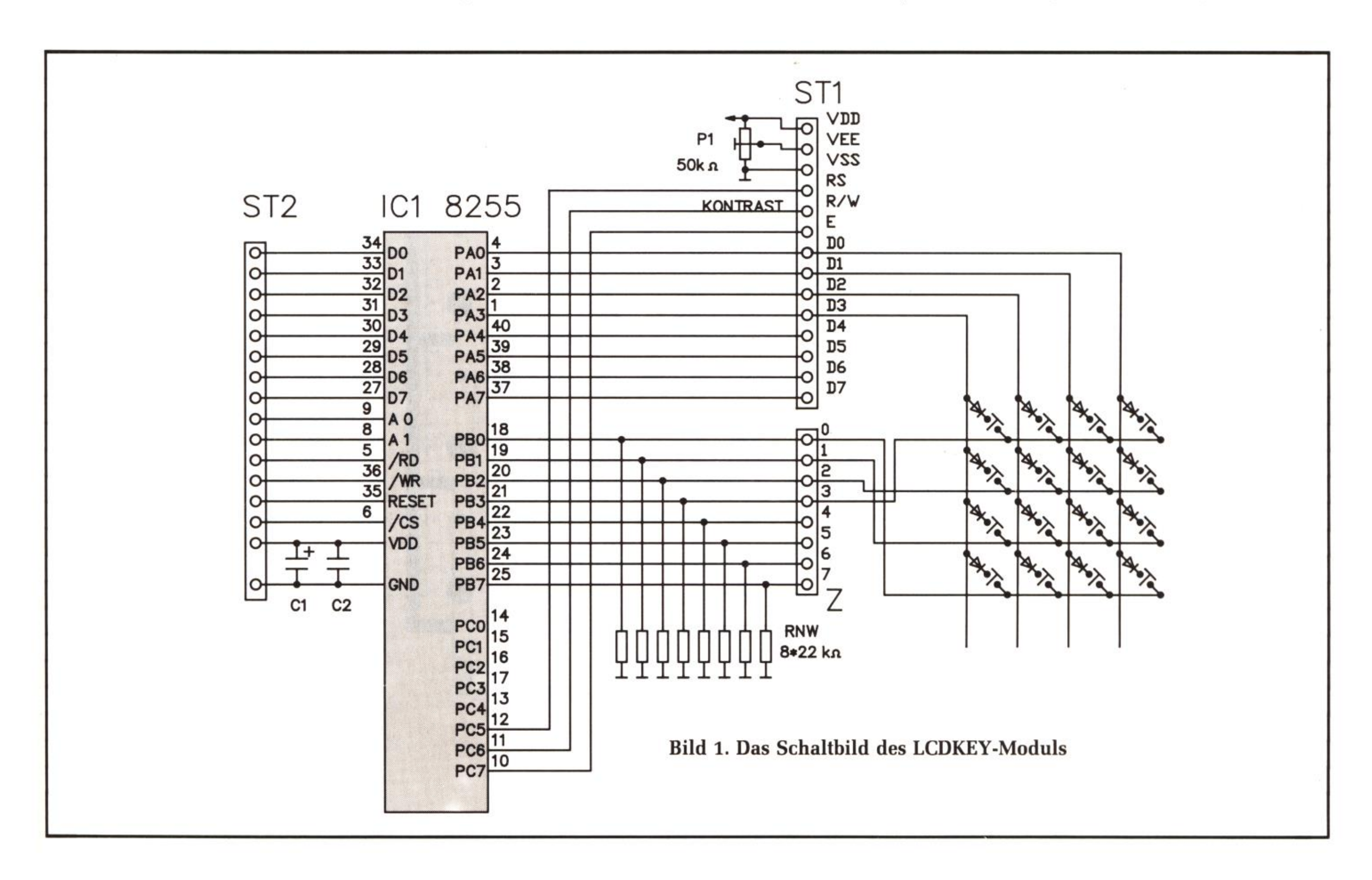


Bild 1. Das Schaltbild des LCDKEY-Moduls

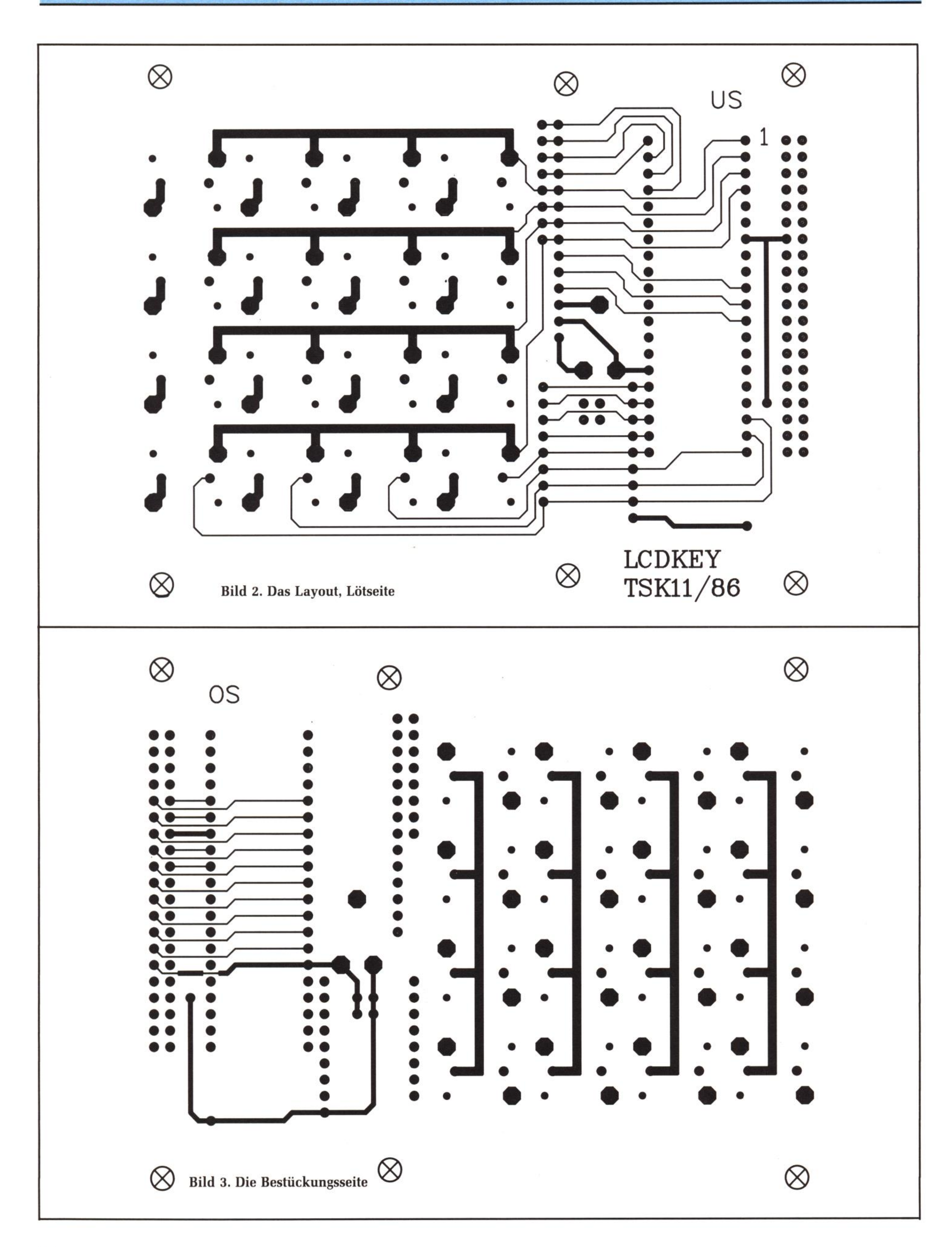


Bild 2. Das Layout, Lötseite

Bild 3. Die Bestückungsseite

Für den PIO-Baustein sollte nach Möglichkeit eine CMOS-Version (82C55, 71055) eingesetzt werden, um den Stromverbrauch so gering wie möglich zu halten. Er beträgt dann nur einige Milliampere (ca. 3 mA).
Die Ansteuerung der Tastenmatrix erfolgt mit Port A (Ausgänge) und Port B (Eingänge). Zur Entkoppelung der einzelnen Tasten untereinander dient jeweils eine Diode an jedem Tastenkontakt. Damit ist es möglich auch Tastenkombinationen von zwei und mehr Tasten auszuwerten (z.B. SHIFT-Taste).
Die Abfrage erfolgt in der Weise, das am Port A jeweils eine Leitung auf „HIGH" gesetzt wird und dann am Port B geprüft wird, ob irgendein Eingang ebenfalls auf „HIGH" liegt.

Aufbau

In *Bild 2 und 3* ist das Layout der zweiseitigen Platine angegeben. *Bild 4* zeigt den Bestückungsplan der Baugruppe. Zur Verwendung einer eigenen Tastatur sind entsprechende Kontakte auf der Platine vorgesehen. Das Tastenfeld kann dazu abgetrennt werden. Bei Verwendung des vorgesehenen Tastenfeldes muß darauf geachtet werden, daß zuerst die Dioden unter den Tastern , dann erst die Taster selbst eingebaut werden. Die restlichen Bauteile nach Stückliste werden von der Bestückungsseite der Platine eingebaut. Vor der Montage des Displays ist die Platine sorgfältig auf Löt- und Bestückungsfehler zu überprüfen. Das LC-Modul wird mechanisch mit kleinen Abstandsbolzen befestigt, erst danach erfolgt der elektrische Anschluß.

Inbetriebnahme

Nach genauer Überprüfung auf Fehler wird die LCDKEY Baugruppe an den Basic-EMUF mit einem 40 poligen Flachbandkabel am PIO-Stecker (ST4) angeschlossen. Die Länge dieses Kabels sollte 30 cm nicht übersteigen. Ein EPROM mit dem angebenen Treiberprogramm muß im Assembler-EPROM-Sokkel (IC 8) des Basic-EMUF eingesteckt sein. Nach dem Einschalten mit angeschlossenem LCDKEY sollte sich der Basic-EMUF wie gewohnt am Terminal melden. Der Kontrasteinsteller auf der Rückseite der Platine wird jetzt so eingestellt, daß die obere Reihe der Anzeige dunkle Kästchen anzeigt (fast rechter Anschlag von P1). Nach der Initialisierung mit CALL 1 ist jetzt ein blinkender Cursor in der oberen linken Ecke sichtbar. Mit dem Befehl „UO1" kann nun die Ausgabe auf das Display umgeleitet werden. Mit dem Befehl „UO0" kann jederzeit wieder auf das Terminal zurückgeschaltet werden.

Universal Input/Output Programm

Ein Bausatz mit Anleitung und universellem Input-/Output-Programm ist beim Elektronikladen in Detmold erhältlich. Die mc-Redaktion verschickt das Listing von UIO auf Anfrage.
Das Ansteuerprogramm für die LCDKEY-Baugruppe belegt nur einige hundert Byte des Eproms. Es wurden noch einige andere Unterprogramme im EPROM untergebracht. Diese dienen hauptsächlich der Ansteuerung verschiedener Peripheriebaugruppen des Basic-EMUFs. Weiterhin wurde ein Unterprogramm hinzugefügt, das die Funktion VALUE (Umwandlung eines Strings in eine Fließkommazahl) durchführt.
Für die jeweiligen Funktionen sind in *Bild 5* verschiedene einfache Testprogramme angegeben, die die Verwendung dieser kleinen Hilfsprogramme zeigen.

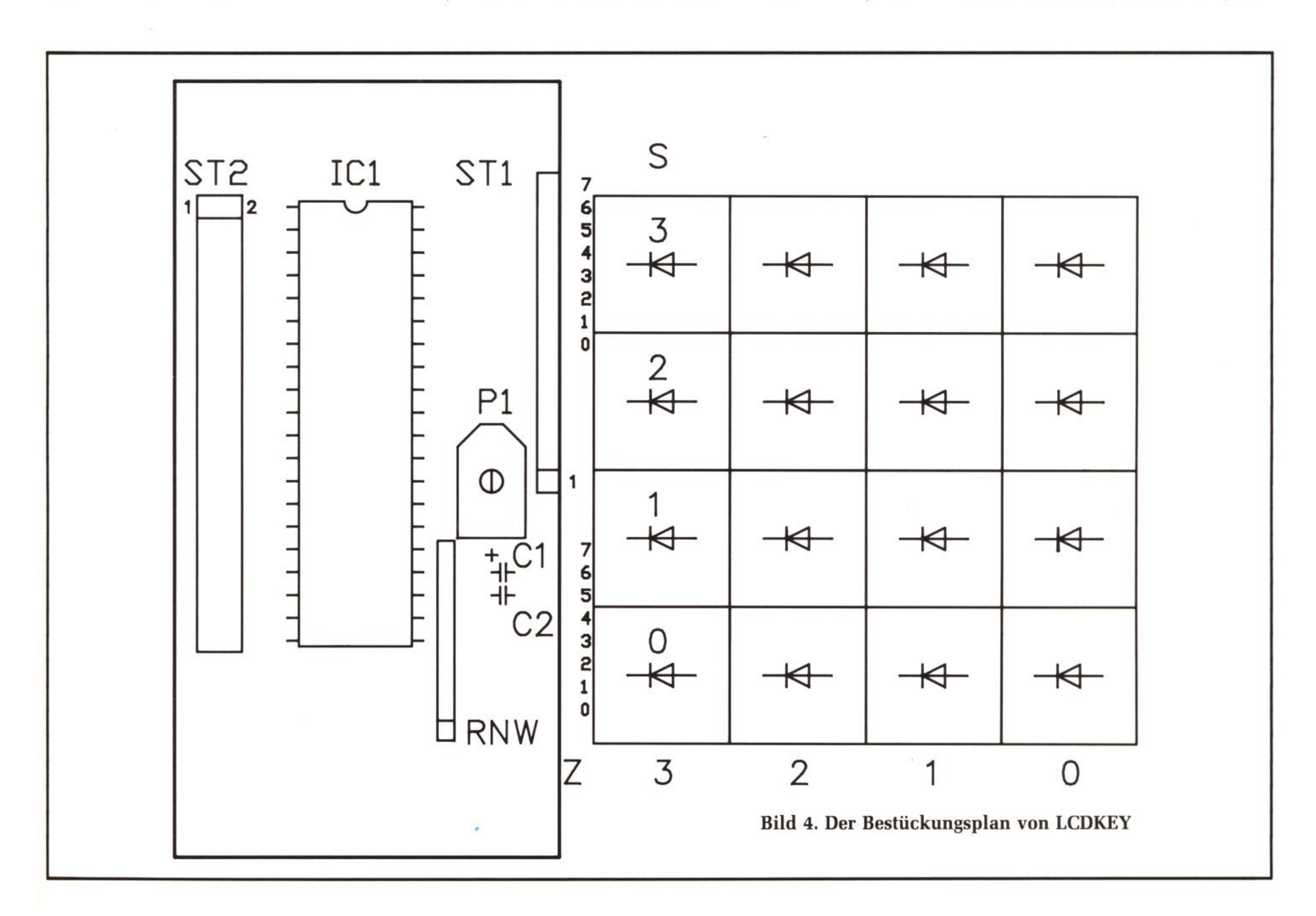


Bild 4. Der Bestückungsplan von LCDKEY

```
Aufruf      NAME        Funtion
CALL 0      DORES       Führt einen absoluten Sprung nach 0h aus.
                        Entspricht in etwa einem Software Reset
CALL 1      LCDINI      Initialisierung LCDKEY
CALL 2      LCDDAT      Ausgabe von [TOS] an das Daten-Register
                        des LCD-Moduls. Dient zur Ansteuerung des
                        programmierbaren Zeichengenerators usw.
CALL 3      LCDCON      Ausgabe von [TOS] an das Control-Register
                        des LC-Moduls. Dient hauptsächlich zur
                        freien CURSOR-Positionierung
CALL 4      ADC         Messen eines AD-Kanals. Kanal in [TOS].
                        Ergebnis wird in [TOS] zurückgeliefert
CALL 5      READTM      Einlesen der Uhrzeit nach $(0).
                        Strings müssen mit min. 17 Zeichen definiert
                        sein (z.B. STRING 100,20)
CALL 6      RDNBAS      Lesen eines 4-Bit Wertes (Nibble) der Uhr RTC58321.
                        Aufruf mit [TOS]=Adresse des Nibbles.
                        Rückkehr mit [TOS]=Daten des Nibbles
CALL 7      WRNBAS      Schreiben eines 4-Bit Wertes (Nibble) der RTC58321.
                        Aufruf mit [TOS-1] = zu schreibender Wert
                                   [TOS]   = Adresse des Nibbles
CALL 8      VALUE       Wandelt $(0) in eine Fließkommazahl
                        Rückkehr [TOS] = Status, bei Fehler<>0
                        [TOS-1] = Fließkommazahl wenn Status=0
CALL 9      FORINI      Initialisierung der Parameter für die Drucker-
                        schnittstelle mit [TOS]. Freigabe des PRINT@,LIST@
                        Treiberprogramms


Programmbeispiele:

10      REM TESTPROGRAMM ADC,LCD
20      PRINT "BEENDEN DURCH BELIEBIGEN TASTENDRUCK"
30      CALL 1 : UO 1 :  REM INITIALISIERUNG UND UMSCHALTEN AUF LCDKEY
40      PRINT CHR(5) :  REM CURSOR AUS
50      PUSH 0 :  CALL 4 :  REM MESSEN KANAL 0
60      POP W :  REM W = MESSWERT
70      PUSH 80H :  CALL 3 :  REM CURSOR 1.ZEICHEN 1.ZEILE POS.
80      PRINT W :  REM AUSGABE DES MESSWERTES
90      FOR I=1 TO 100 :  NEXT I :  REM WARTESCHLEIFE
100     A=GET :  IF A=0 THEN 50
110     UO 0

10      REM TESTPROGRAMM UHR
20      STRING 106,20 :  REM STRINGRESERVIERUNG 5*20 BYTE
30      PRINT "ZUM STELLEN DER UHR MUSS JUMPER J5 GESCHLOSSEN SEIN"
40      PRINT
50      INPUT "WELCHES NIBBLE SOLL BESCHRIEBEN WERDEN ? ",AD
60      PUSH AD :  CALL 6 :  REM LESEN RTC
70      POP DA :  REM WERT HOLEN
80      PRINT "MOMENTANER WERT =",DA,
90      INPUT "NEUER WERT ? ",DA
100     PUSH DA,AD :  CALL 7 :  REM SCHREIBEN DES NEUEN WERTES
110     CALL 5 :  PRINT $(0) :  REM AUSGABE DER UHRZEIT
120     PRINT "WEITERMACHEN (J/N) ? ",
130     A=GET :  IF A=0 THEN 130 :  REM WARTEN AUF TASTENDRUCK
140     IF A=ASC(J) THEN 40

10      REM TESTPROGRAMM UHR,LCD
20      STRING 106,20 :  REM STRINGRESERVIERUNG 5*20 BYTE
30      PRINT "BEENDEN DURCH BELIEBIGEN TASTENDRUCK"
40      CALL 1 :  UO 1 :  REM INITIALISIERUNG UND UMSCHALTEN AUF LCDKEY
50      PRINT. CHR(5) :  REM CURSOR AUS
60      CALL 5 :  REM LESEN DER UHRZEIT NACH $(0)
70      PUSH 80H :  CALL 3 :  REM CURSOR 1.ZEICHEN 1.ZEILE POS.
80      FOR I=1 TO 9 :  PRINT CHR(ASC($(0),I)), :  NEXT I :  REM DATUM
90      PUSH 0C0H :  CALL 3 :  REM CURSOR 1.ZEICHEN 2.ZEILE POS.
100     FOR I=10 TO 18 :  PRINT CHR(ASC($(0),I)), :  NEXT I :  REM ZEIT
110     A=GET :  IF A=0 THEN 50 :  REM WARTEN AUF TASTENDRUCK
120     UO 0

10      REM TESTPROGRAMM TASTENFELD
20      STRING 106,20 :  REM RESERVIERUNG STRINGSPEICHERPLATZ
30      CALL 1 :  REM INITIALISIERUNG LCDKEY
40      PRINT "GEBEN SIE EINEN STRING AM 16ER-TASTENFELD EIN"
50      UI 1 :  REM UMSCHALTEN TASTENFELD
60      INPUT $(0)
70      UI 0 :  REM ZURUECKSCHALTEN AUF TERMINAL
80      PRINT $(0)

10      REM TEST VALUE FUNKTION
20      STRING 100,10
30      INPUT $(0)
40      CALL 8
50      POP ST
60      IF ST<>0 THEN  PRINT "Falsche Eingabe! Bitte wiederholen" :  GOTO 30
70      POP Z
80      PRINT "Die eingegebene Zahl ist ",Z
```

Bild 5. Aufzählung der Funktionsunterprogramme

Ansteuerung des LCDKEY

Zur Initialisierung muß vor Gebrauch der Ausgabe-Routinen das Display initialisiert werden. Dies erfolgt mit dem Befehl „CALL 1". Dieser Befehl braucht nur einmal in einem Programm durchgeführt zu werden. Die normale Ausgabe des „PRINT" Statements, kann nach Ausführung von „UO1" auf das LC-Modul umgeleitet werden. Jederzeit kann jedoch mit „UO0" wieder auf die serielle Terminalschnittstelle zurückgeschaltet werden.

An Steuerzeichen werden folgende ASCII-Werte ausgewertet:

04H blinkender Cursor ein
05H Cursor aus
08H ein Zeichen nach links (max. bis Zeilenanfang)
0AH neue Zeile (Wechsel zwischen 1. und 2. Zeile, kein Scroll)
0DH Cursor an Zeilenanfang

Andere Steuercodes sind zur Zeit nicht implementiert und werden ignoriert (sonstige ASCII-Werte zwischen 0 und 1FH). Zur Ansteuerung der internen Register des LC-Moduls dienen die Unterprogramme „CALL 2" und „CALL 3".

Das Tastenfeld

Das Unterprogramm zur Ansteuerung des Tastenfeldes auf dem LCDKEY-Modul ist mit einer SHIFT- und CONTROL-Funktion ausgestattet. Dabei ist die Tastenposition dieser SHIFT- und CONTROL-Tasten frei programmierbar. Es kann für jede Tastenkombination (Normal, Shift, Control, Shift+Control) ein Eintrag in einer Tabelle abgelegt werden, und damit auch der jeweilige Tastencode. Die derzeitige Belegung kann *Bild 6* entnommen werden.

VALUE

Diese Routine dient der Umwandlung eines Strings in eine Fließkommazahl. Bei INPUT-Befehlen am LCDKEY wird bei Eingabe einer falschen Fließkommazahl die Meldung „TRY AGAIN" ausgegeben und dabei das Display gelöscht. Um dies zu vermeiden, kann jetzt ein String eingegeben werden und danach per Programm gewandelt werden.

Gleichzeitig wird eine Prüfung des Strings vorgenommen, wodurch Eingabefehler in der Software abgefangen werden können.

! A N 7	" B O 8	# C P 9	shift
$ D Q 4	% E R 5	& F S 6	control
(G T 1	) H U 2	= I V 3	- J W +
? K X del	* L Y 0	/ M Z .	CR

shift+control   shift
control   normal

Bild 6. Die gegenwärtige Belegung der Funktionstasten

PRINT@, LIST@ ... Routinen

Beim Anschluß eines seriellen Druckers und Ansteuerung mit LIST#, und PRINT# ist es bisher erforderlich, die Schnittstelle mit einer niedrigen Baudrate zu betreiben, da im internen Druckerprogramm keine Überprüfung eines Busy-Signals vorgesehen war. Dies wurde durch Verwendung des PRINT@ Treibers ermöglicht. Außerdem ist es bei diesem Treiberprogramm vorgesehen, die Datenübertragungsparameter einzustellen. Dies geschieht mit Hilfe eines 8-Bit Datenbytes, das eine Codierung nach *Bild 7* besitzt.

Zur Initialisierung der Schnittstelle muß der Wert des Format-Bytes auf den Argument-Stack gelegt werden, danach erfolgt ein Aufruf mit CALL 9. Das kann auch während des Programmlaufs durchgeführt werden. Nach dem CALL 9-Aufruf werden die Argumente der PRINT@, LIST@ usw. Kommandos an die serielle Druckerschnittstelle übergeben (*Bild 8*). Eine Überprüfung der TIMEOUT-Funktion ist aus dem Basic möglich durch Zugriff auf die interne Speicherstelle 1AH. Dieser Zustand kann mit „DBY(1AH).and.80H" geprüft werden. Ist dieser Wert<>0, liegt ein TIMEOUT-Fehler vor.

Unterprogramm-Aufruf

Der Aufruf der Unterprogramme erfolgt jeweils mit einem CALL N-Statement. N ist dabei die jeweilige Funktionsnummer (Bild 5). Eventuell notwendige Daten werden dabei auf den Argument-Stack übergeben. [TOS] ist hierbei die Zahl auf dem Argument-Stack, [TOS-1] ... [TOS-n] die darunterliegenden Zahlen.

```
Bit   7    6    5    4    3    2    1    0
Abk.  TO   X    BSY  S    P1   PO   D1   DO

Bedeutung:
TO    TimeOut-Flag    Wird beim Ansprechen der Routine
                      das BUSY-Signal innerhalb von etwa 10 s nicht aktiv,
                      wird dieses Bit gesetzt und die weitere Ausgabe
                      unterdrückt, bis das Bit wieder gelöscht ist.
X                     z.Zt. nicht benutzt
BSY                   0=aktiv low Busy / 1=aktiv High Busy
S                     0=1 Stopbit / 1=2 Stopbit
P1,PO                 00=keine 01=ODD 10=EVEN 11=MARK Parität
D1,DO                 00=5 01=6 10=7 11=8 Datenbits
```

Bild 7. Die Bedeutung der Bits im Schnittstellensteuer-Byte

```
Datenformat  =  4800 Baud,8 Bit,  keine Parität,  1 Stopbit,
                Busy über  CTS Leitung (Busy=aktiv high)

            TO   X    BSY  S    P1   PO   D1   DO
FORMAT =    0    0    1    0    0    0    1    1    = 023H

Eingabe im Kommando-Modus:
>PUSH 23H
>CALL 9
>BAUD 4800
>LIST@
READY
>
```

Bild 8. Ein Listing wird mit 4800 Baud abgerufen

```
********************************* ! WICHTIG *********************************
Die Dioden und Taster sind von der Lötseite her zu bestücken.  Dioden zuerst
einlöten.
*****************************************************************************

Stückliste
DISP       SHARP 16255, HITACHI LM016L
IC1        8255, 71055 o.ä.
D          16 Dioden 1N4148 o.ä.
T          4 Siemens-Tastenstreifen (je 4 Taster)
RNW1       Netzwerk 9pol, SIL 8*10 kOhm (22 kOhm)
P1         Poti 20 kOhm bis 50 kOhm
C1         10 mikroF/10 V, Tantal
C2         100 nF Vielschicht
ST2        Pfostensteckerleiste 2reihig 40pol.

Steckerbelegung

ST1 Buchsenleiste für LC-Modul
1    GND
2    +5 V
3    KONTRAST-SPANNUNG 0-5 V
4    RS
5    R/W
6    E
7    DO
8    D1
9    D2
10   D3
11   D4
12   D5
13   D6
14   D7


ST2 Anschluß an Basic-Emuf PIO-Stecker

1                        2
3                        4
5                        6
7                        8
9    /WR                 10   /RD
11   RESET               12   /PIOST
13   D 0                 14   GND
15   D 1                 16   A1
17   D 2                 18   AO
19   D 3                 20
21   D 4                 22
23   D 5                 24
25   D 6                 26
27   D 7                 28
29   +5 V                30
31                       32
33                       34
35                       36
37                       38
39                       40
```

Bild 9. Das ist die Stückliste

Erhard Scherer

Der EMUF08

Ein starker Einplatinencomputer

Seit 1981, als der erste EMUF, eine Einfach-Euro-Karte mit 6504 CPU, von Herwig Feichtinger vorgestellt wurde, hat das Einplatinen-Computer-Konzept tausendfach Verwendung im privaten, industriellen und universitären Bereich gefunden. Seit der Zeit des 6504-EMUF ist die technische Entwicklung im Bereich der Mikroelektronik mit atemberaubender Geschwindigkeit vorwärts gestürmt. Die Integrationsdichte der Chips stieg um ein Vielfaches, für Speichergrößen, die noch vor einigen Jahren ganze Karten belegten, genügt heute ein einziger Baustein. Parallel zur Steigerung der Integrationsdichte ist ein deutlicher Preisverfall zu beobachten, der einen 68008-EMUF erschwinglich macht.

Heute kostet beispielsweise eine 68000-CPU (16 Bit) etwa genauso viel, wie vor wenigen Jahren ein Z80 (8 Bit). Das Aufkommen der 16-Bit-CPUs brachte Architekturen in den Mikrocomputerbereich, wie sie vorher nur bei sehr teuren Minicomputern, die zudem ganze Schaltschränke füllten, anzutreffen waren. Die 68000er Familie ist dafür ein sehr gutes Beispiel. Kenner der PDP11 entdecken sehr viele Parallelen zwischen beiden Rechnertypen, bezüglich des Registersatzes, der Maschinenbefehle usw.

Der neue EMUF vereinigt nun die Preiswürdigkeit der 8-Bit-EMUFs mit Architekturmerkmalen von Minicomputern. Das Herz des EMUF08 ist ein 68008. Dank des 8 Bit breiten äußeren Datenbusses (intern im Chip selbst sind die Busse 16 Bit breit) ist der 68008 im platzsparenden 48-Pin-Gehäuse untergebracht. Die 680xx-Familie hat sich inzwischen einen festen Platz als Leistungsträger in industriellen Anwendungen erobert; aber auch in Personal Computern wie dem Macintosh, dem Atari 520 oder dem Amiga gibt ein 68000er den Ton an. Die Verbreitung und Bekanntheit des 68000 läßt den Wunsch nach einem preiswerten Einplatinencomputer mit 68000-CPU für Steuer-, Meß- und Regelaufgaben aufkommen. Der EMUF08 erfüllt diesen Wunsch.

Die extreme Registerstruktur sowie der leistungsfähige Maschinenbefehlssatz des 68000 machen diesen EMUF für Aufgaben fit, bei denen Höchstleistungen gefordert sind. Der 68008 besitzt 8 Datenregister mit 32 Bit Breite (dies führt übrigens zu der Hochstapelei mancher Platinenhersteller, die Systeme mit dem 68000 als 32-Bit-Systeme verkaufen) und 8 Adreßregister (32 Bit breit), wobei A7 der Stackpointer ist. Es gibt zwei Betriebsmodi für die CPU: den User Mode, der mit einigen Einschränkungen verbunden ist, und den Supervisor Mode. Für beide Modi kann ein eigener Stackpointer (A7' bzw. A7) eingerichtet werden. Programme für den EMUF werden vorzugsweise im Supervisor Mode gefahren, da die Benutzung des User Mode nur in Multitasking/Multiuser-Systemen sinnvoll ist und zudem von der Hardware unterstützt werden sollte.

EMUF08 und Software

Die Erstellung von Software, in Assembler oder einer höheren Programmiersprache (z. B. „C") für dem EMUF kann auf den eingangs erwähnten Maschinen vom Apple, Atari, Commodore erfolgen – oder natürlich auch auf dem mc-68000-Computer, dem NDR-Klein-Computer, Hermann-20 von MTC Berlin (eine echte 32-Bit-Maschine mit 68020 CPU) oder dem c't-68000. Der Vorteil, sowohl auf dem Entwicklungssystem als auch auf der Zielhardware (in unserem Fall der EMUF) eine CPU vom selben Typ zu haben, liegt darin, daß die Assemblerprogrammierung nur einmal erlernt werden und auch während der Arbeit nicht ständig umgedacht werden muß. Ein weiterer Gesichtspunkt ist, daß Compiler, Assembler usw. nur einmal gekauft werden müssen. Wer einmal in der Assemblerprogrammierung den Schritt von den guten alten 8-Bit-CPUs zum 68000 getan hat, ist so verwöhnt, daß ihn ein Z80, 8085 oder 6800 nicht mehr reizen können. Ganzzahlarithmetik, mit 32 Bit breiten Zahlen, gibt's beim EMUF08 serienmäßig eingebaut und ohne Aufpreis. Die 68008-CPU besitzt Addition, Subtraktion, Multiplikation (16×16 Bit, 32 Bit Ergebnis) und Division (32/16 Bit, 16 Bit Ergebnis, 16 Bit Rest) bereits als Maschinenbefehle.

Beispiel: DIVS D1,D0

Das ist die vorzeichenbehaftete Division, mit dem Ergebnis in den niederwertigen 16 Bit von D0, dem Rest in den höherwertigen 16 Bit von D0. Diese Fähigkeit zum Jonglieren mit Ganzzahlen eröffnet dem EMUF08 Anwendungen, bei denen 8-Bit-CPUs aus Geschwindigkeitsgründen passen müssen, denn Integerarithmetik in Software ist langsamer als es die gewissermaßen in Hardware gegossenen Mikroprogramme des 68000 sind. Wer von den algorithmischen Höhen nicht ganz bis auf den Grund der Assemblerprogrammierung herabsteigen möchte, kann natürlich auch mit einer höheren Programmiersprache arbeiten. Die im Vergleich zu 8-Bit-CPUs größere Leistungsfähigkeit erlaubt es, mit dem EMUF08 Aufgaben in einer höheren Programmiersprache zu lösen, für die sonst aus Geschwindigkeitsgründen nur Assembler in Frage käme. Programme in höheren Programmiersprachen benötigen nicht nur mehr Platz als gleichwertige Assemblerprogramme, sondern sind in ihrer Ausführung auch langsamer. Höhere Programmiersprachen haben natürlich auch Vorteile: der Quellcode ist kürzer als in Assembler, die Programme sind im allgemeinen lesbarer und somit leichter zu pflegen. Nicht zuletzt sei die leichtere Übertragbarkeit auf eine andere Hardware erwähnt, für die z. B. „C" bekannt ist.

Die Hardware des EMUF

Der EMUF08 ist auf einer Einfach-Euro-Karte (100× 160 mm) untergebracht, ein Viertel der Platine ist Lochrasterfeld, drei Viertel belegt die Elektronik. Sie besteht (*Bild 1*) aus der 68008-CPU, RAM (8 KByte bis 32 KByte), EPROM (8 KByte bis 64 KByte), einigen TTL Bausteinen, sowie wahlweise einer VIA

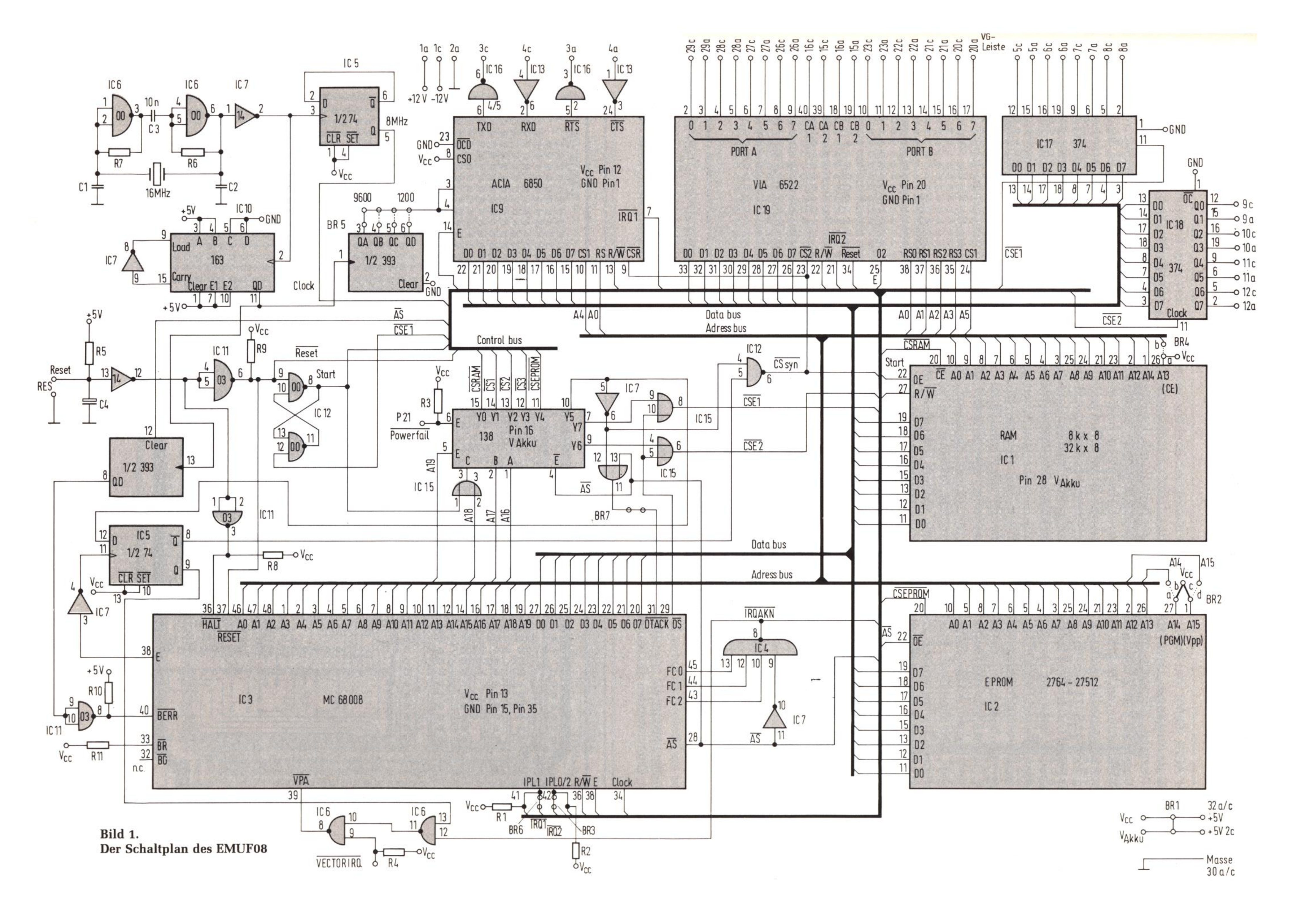


Bild 1.
Der Schaltplan des EMUF08

(6522), einer ACIA (6850) und zwei 8-Bit-Latches als Ausgängen. Damit stehen 18 Ein-/Ausgabe-, 2 Eingabe- und 16 Ausgabeleitungen zur Verfügung, insgesamt 36 parallele Leitungen. Die 16 Ausgänge der beiden Latches können, bei Verwendung von 74AS374 Bausteinen, 64 mA (bei 0-Pegel) Last treiben. Relais, LED Anzeigen usw. kann man also direkt anschließen. Die parallelen I/O Leitungen, die serielle Schnittstelle und die Stromversorgung führen auf eine 64polige VG-Leiste. Der „Anbau" von Zusatzhardware wie Anzeigen, Analog-Digital-Wandler, Relais ... ist über diesen Stekker durchzuführen. Die wichtigsten Anschlüsse der CPU und einige andere Signale führen auf eine Doppelreihe des Lochrasterfeldes (*Bild 2* zeigt die Lötseite der Platine, *Bild 3* die Bestückungsseite und *Bild 4* den Bestückungsplan).
Wer mag, kann zum Beispiel über eine Pfostensteckverbindung Hardware aus eigener Entwicklung anschließen. Dafür steht ihm der Adreßraum ab hexadezimal 80000 bis FFFFF zur Verfügung. Dies sind 512 KByte.

Hinweise für den Aufbau

Für das Löten und Bestücken hat sich folgende Vorgehensweise als zweckmäßig erwiesen: Zuerst werden die Bauteile mit der niedrigsten Bauhöhe eingelötet, dann die mit der nächsthöheren, usw. Da immer Bauteile mit derselben Bauhöhe zu löten sind, können die Bauteile mit einer starken Pappe oder etwas ähnlichem gegen Herausfallen gesichert werden. Dieser Sandwich wird dann um 180 Grad gedreht, und die Bauteileanschlüsse können verlötet werden. Dabei stören die im vorherigen Arbeitsgang eingelöteten Bauteile nicht, da sie ja schon fest mit der Platine verbunden sind und zudem eine geringere Bauhöhe haben. Die „teuren" Bausteine, wie z. B. den MC68008, lötet man am besten nicht direkt ein, sondern stekt sie auf einen Sockel. Das Sockeln ist auch bei RAM und EPROM zweckmäßig, beim EPROM um das Austauschen bei einer Softwareänderung einfach zu gestalten, beim RAM um den Wechsel von den 8-KByte-RAMs auf die 32-KByte-RAMs ohne Lötkolben durchführen zu können.
Tabelle 1 zeigt die Stückliste.

Der EMUF08 muß, um funktionsfähig zu sein, nicht unbedingt voll bestückt werden. Folgende Bausteine können erst einmal weggelassen werden: RS232-Treiber (IC13, IC16), ACIA 6850, VIA 6522 und die Latches 74AS374. CPU, EPROM, RAM und Decodierlogik reichen für erste Funktionstests aus.

Der Anschluß der Versorgungspannung erfolgt über die 64polige VG-Leiste (siehe *Tabelle 2*, Steckerbelegung). Über diese Leiste können auch anwenderspezifische Schaltungen an den EMUF angesteckt werden, dies werden in der Regel Schaltungen sein, die VIA, ACIA oder die Latches benutzen. Der EMUF08 benötigt 5 V Versorgungsspannung, das Netzteil sollte für 1 A gut sein, um etwas Reserve für Zusatzschaltungen zu haben. Bei Benutzung der ACIA und der RS232-Treiber werden noch +12 V und −12 V benötigt (je ca. 50mA).

Die Anpassung an Bestückungsvarianten

Die Platine des EMUF08 ist mit einer Reihe von Steckbrücken ausgestattet, die eine einfache Anpassung an unterschiedliche Bestückungsvarianten und Betriebsarten ermöglichen. Bevor die Versorgungsspannung angelegt wird, sollte eine allgemeine Sichtkontrolle auf Löt- und Bestückungsfehler sowie auf die richtige Lage der Steckbrücken erfolgen (*Tabelle 3*).
Ein erster Funktionstest des EMUF kann mit einem einfachen Terminal (auch mit Hostcomputer) mit Hilfe von MONI-E8 (im EPROM) erfolgen. Falls beides nicht zur Verfügung steht, kann das untenste-

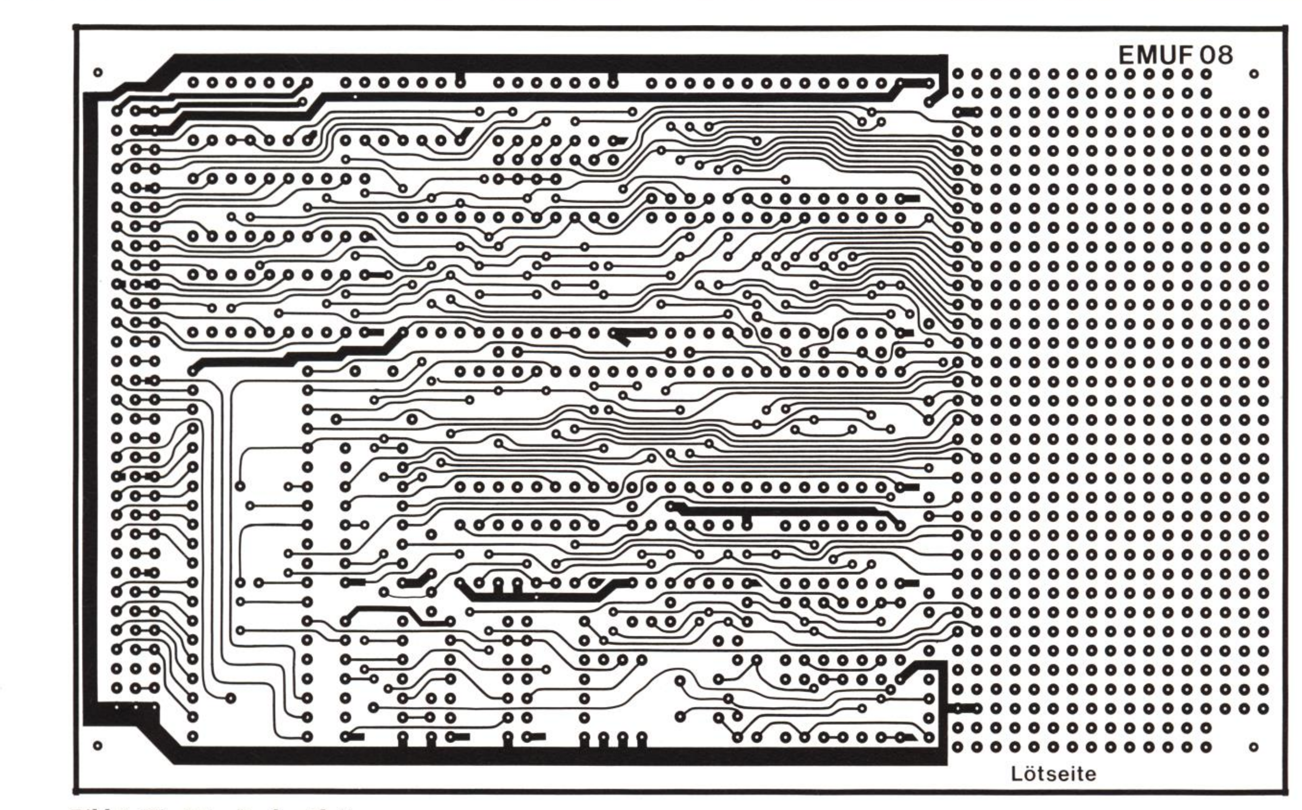


Bild 2. Die Lötseite der Platine

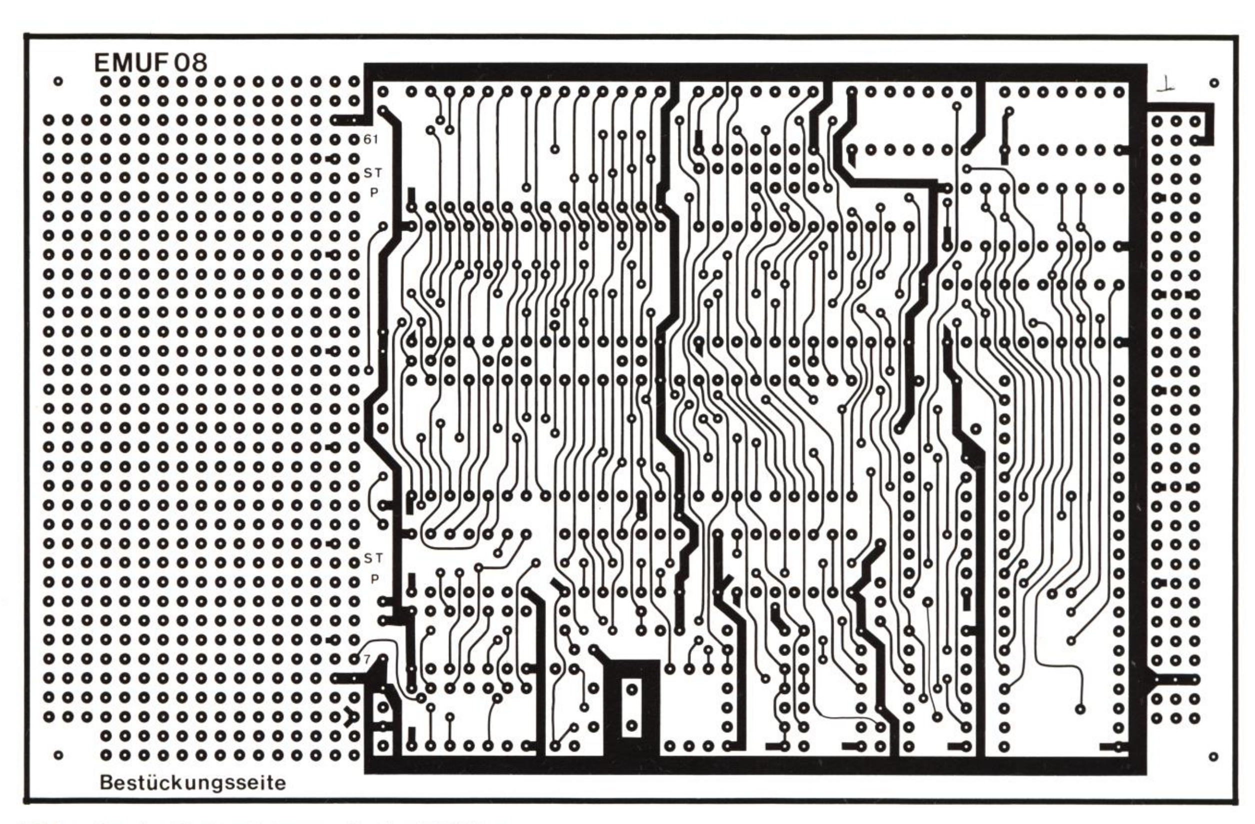


Bild 3. Das ist die Bestückungsseite des EMUF08

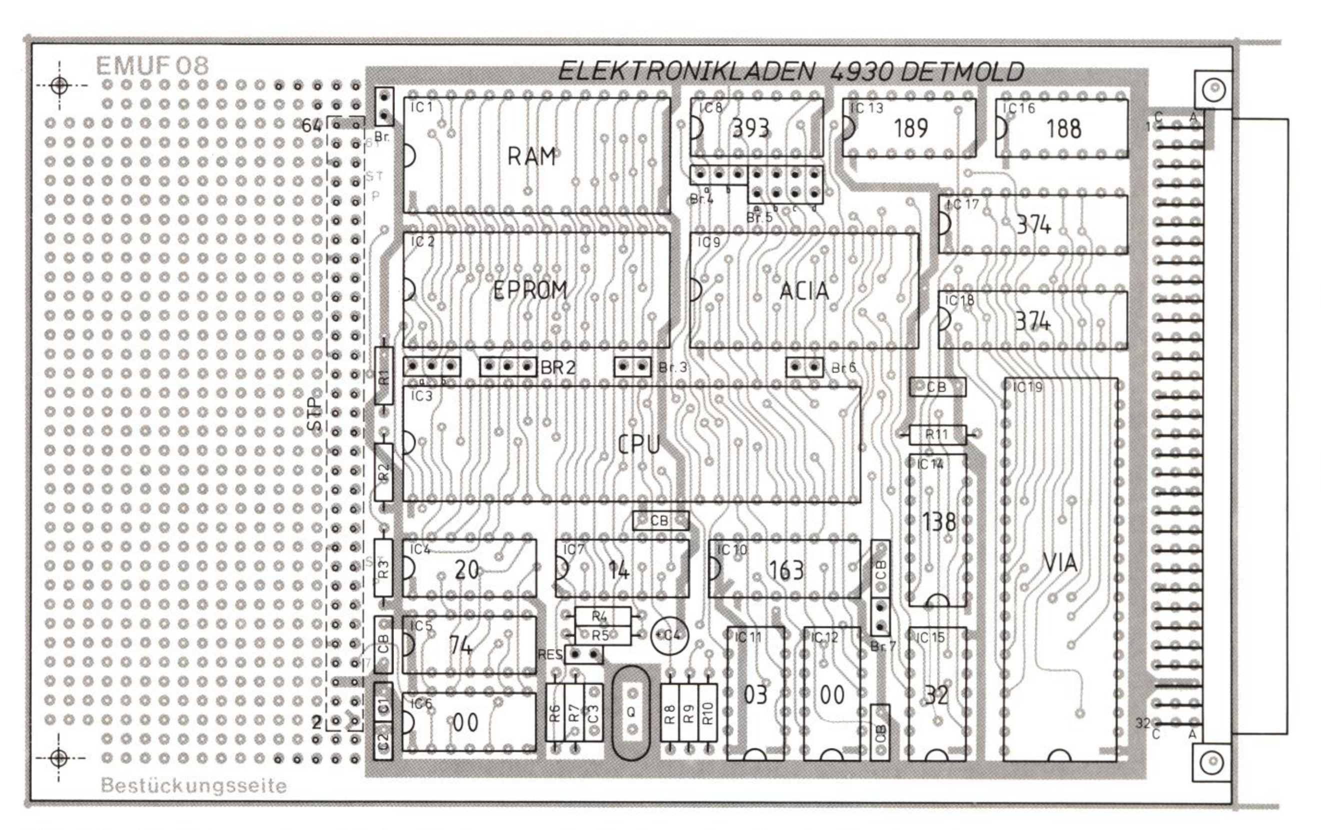


Bild 4. Der Bestückungsplan zeigt eine Mischung aus bewährten und modernsten Bauelementen

Tabelle 1: Stückliste zum EMUF

IC1	RAM	8 oder 32 KByte (max. 250 ns)
IC2	EPROM	8...64 KByte (max. 250 ns)
IC3	CPU	MC68008, 8 MHz
IC4	LS/HCT	7420
IC5	LS/HCT	7474
IC6	LS	7400
IC7	LS/HCT	7414
IC8	LS/HCT	74393
IC9	ACIA	6850, 1 MHz
IC10	LS/HCT	74163
IC11	LS/HCT	7403
IC12	LS/HCT	7400
IC13		75189 o. 1489
IC14	LS/HCT	74138
IC15	LS/HCT	7432
IC16		75188 o. 1488
IC17	S/AS	74374
IC18	S/AS	74374
IC19	VIA	6522, 1 MHz
R1	¼ Watt, 5 %	1 kΩ
R2		1 kΩ
R3		1 kΩ
R4		1 kΩ
R5		22 kΩ
R6		3,3 kΩ
R7		1,5 kΩ
R8		1 kΩ
R9		1 kΩ
R10		1 kΩ
R11		1 kΩ
CB	Vielschicht	0,1 µF
C1	Scheibe	22 pF
C2	Scheibe	22 pF
C3	Scheibe	10 nF
C4	Elko	10 µF
Q	Quarz	16 MHz

Tabelle 2: Belegung der VG-Leiste

	a	b	c		a	b	c
1	+12 V	−12 V	−12 V	17	–		
2	GND	Akku	Akku	18	–	–	–
3	RTS	TXD	TXD	19	–	–	–
4	CTS	RXD	RXD	20	PB7	PB6	PB6
5	L1-1	L1-0	L1-0	21	PB5	PB4	PB4
6	L1-3	L1-2	L1-2	22	PB3	PB2	PB2
7	L1-5	L1-4	L1-4	23	PB1	PB0	PB0
8	L1-7	L1-6	L1-6	24	–	–	–
9	L2-1	L2-0	L2-0	25	–	–	–
10	L2-3	L2-2	L2-2	26	PA7	PA6	PA6
11	L2-5	L2-4	L2-4	27	PA5	PA4	PA4
12	L2-7	L2-6	L2-6	28	PA3	PA2	PA2
13	–	–	–	29	PA1	PA0	PA0
14	–	–	–	30	GND	GND	GND
15	CB2	CA2	CA2	31	–	–	–
16	CB1	CA1	CA1	32	+5 V	+5 V	+5 V

hende Testprogramm (*Tabelle 4*), das in ein EPROM „gebrannt“ wird hilfreich sein.
Mit Hilfe eines Oszillografen läßt sich durch Darstellung des Verlaufs der CPU-Signale AS∗ und eventuell der Datenleitungen D0–D7 die Funktion des EMUF nachweisen – oder aber die Fehlerstelle kann systematisch eingekreist werden. In ganz schwierigen Fällen kann ein periodischer Reset bei der Fehlersuche helfen (siehe mc Heft 3/86, „Das doppelte Lottchen“, Seite 55). Nach dem Reset ist das Start-Flip-Flop, gebildet aus zwei NAND-Gattern (IC12) gekippt, so daß das Signal „Start“ 1-Pegel hat. Über ein OR-Gatter (IC15) wird der Adreßdekodierlogik vorgegaukelt, die Adreßleitung A18 sei auf 1-Pegel. Die 68008-CPU liest nach dem Reset von der Adresse 00000 den Supervisor Stack Pointer und von der Adresse 00004 den Anfangsstand des Programmzählers. Da A18 an der Dekodierlogik aber zwangsweise „1“ ist, wird dieser Zugriff auf die Adressen 40000 bzw. 40004 umgelenkt, und somit werden die Anfangswerte aus dem EPROM, das ja an Adresse 40000 liegt, gelesen. Für alle EPROMs, die für den EMUF erstellt werden gilt die Sequenz aus *Tabelle 5*, die auf den ersten Adressen des EPROMs steht.

Der EMUF kann zum Speicherriesen werden

An Stecker P (in der Nähe des Lochrasterfeldes) liegen die wichtigsten Signalleitungen der CPU. Erweiterungen, für die das Lochrasterfeld zu klein ist oder für die eine eigene Platine gemacht wird, können über diesen Stecker erfolgen.

Dazu lötet man am besten eine 64polige Federleiste in schmaler Ausführung mit den Anschlußreihen a und b ein. Eine Erweiterung über Stecker P könnte z. B. eine RAM Karte mit einem halben Megabyte sein, für Anwendungen bei denen ein großer Datenpuffer nötig ist.

Die Adreßdekodierung des EMUF ist sehr einfach gehalten, im Prinzip erle-

Tabelle 3: Steckbrücken

Steckbrücke	Funktion	Grundeinstellung
BR1	Verbindung von V_{CC} und V_{Akku}	gesteckt
BR2	EPROM-Typ 2764	b, c gesteckt
	27128	b, c gesteckt
	27256	a, c gesteckt
	27512	a, d gesteckt
BR3	IRQ2 (VIA) auf IPLO/2 68008	gesteckt
BR4	RAM CE/A13	8 KByte a gesteckt
		32 KByte b gesteckt
BR5	Baudrate ACIA	9600 a
		4800 b
		2400 c
		1200 d
BR6	IRQ1 (ACIA) auf IPL1 68008	gesteckt
BR7	DTACK∗ Erzeugung on/off board	gesteckt

Tabelle 4: Testprogramm

EPROM-Adresse	OPCODE	
0000	0000 0480	Stackpointer
0004	0004 0008	Start Program Counter
0008	60FE	Branch auf sich selbst

Tabelle 5: Die Startinformationen

EPROM-Adresse	OPCODE	
0000	0000 0480	Anfangs-SSP
0004	0004 0008	Anfangs-PC
0008	4A39 0007 0000	Start-Flipflop kippen
000E	Hier beginnt das Anwenderprogramm	

digt ein Dekodierer vom Typ 74138 (IC14) die ganze Arbeit. Der Adreßraum von 00000 bis 7FFFF wird in acht 64-KByte-Blöcke aufgeteilt *Tabelle 6*. Um eine einfache Decodierschaltung zu erreichen, gehen wir mit dem Adreßraum verschwenderisch um, andererseits ist ein Adreßraum von einem Megabyte für einen EMUF nicht gerade wenig.

Die serielle Schnittstelle

Die beiden synchronen Bausteine ACIA und VIA werden über das Flip-Flop 7474 (IC5) auf den Prozessor aufsynchronisiert. Dies ist eine Methode, die es erlaubt, so bekannte Bausteine wie 6522 und 6850 einzusetzen. Anstelle dieser „Veteranen" hätte man auch etwas Moderneres nehmen können, nur leider ist es so, daß sich viele Programmierer noch vor dem komplizierten Innenleben dieser Bausteine „fürchten". Folglich wurde der EMUF mit Beliebt-, Bewährtem ausgestattet. ACIA und VIA können Interrupts auslösen, der Interruptausgang der VIA geht auf den Interrupteingang IPL0/2 der CPU und der der ACIA auf IPL1. Dies bedeutet für den Programmierer, daß die autovektorisierten Interrupts 5 und 7 von der VIA ausgelöst werden, während der Vektor 2 von der ACIA angestoßen wird. IRQ1 und IRQ2 gehen über Steckbrücken auf IPL0/2 und IPL1.

Wer möchte, kann diese Brücken trennen und über Stecker P eine eigene Logik für vektorisierte Interrupts aufbauen.

Die Taktrate für die serielle Schnittstelle (ACIA) wird aus dem 16-MHz-Takt des EMUF abgeleitet. Zuerst werden die 16 MHz durch einen Teiler mit dem Teilungsverhältnis 13 geschickt, 74163 (IC10) und dann nochmal durch 2, 4, 8 und 16 geteilt; über ein Feld von Steckbrücken kann die gewünschte Baudrate eingestellt werden (BR5). Für 9600 Baud bedeutet dies, daß der ACIA 16.000.000 : (13 × 2) = 615.384 Hz als Takt angeboten werden, eine Zahl die 64 mal zu groß ist. Glücklicherweise hat die ACIA noch einen internen Teiler der dann einfach per Software auf ein Teilungsverhältnis von 64 eingestellt wird, und schon stimmt unsere Baudrate. Über die Steckbrücke BR5 (1200 Baud) und ein Teilungsverhältnis von 16 in der ACIA, kann z. B. eine Baudrate von 19 200 erreicht werden.

EMUF – auch mit Gedächtnis

Falls es gewünscht wird, daß der Inhalt des RAMs bei Ausfall der Stromversorgung erhalten werden soll, so sind dafür schon Vorbereitungen getroffen. Der Adreßdekodierbaustein (IC14) kann über ein Powerfail-Signal gesperrt wer-

Tabelle 6: Die Adressenbelegung des EMUF

Adresse	Belegung	
00000 0FFFF	RAM	
10000 1FFFF	frei	(CS* vorhanden)
20000 2FFFF	frei	(CS* vorhanden)
30000 3FFFF	frei	(CS* vorhanden)
40000 4FFFF	EPROM	
50000 5000F	frei	
50010	ACIA	Daten-Register
50011		Status-Register
50012		Befehls-Register
50013		Steuer-Register
50020	VIA	Daten-Register B (ORB/IRB)
50021		Daten-Register A (ORA/IRA)
50022		Datenrichtungsregister B
50023		Datenrichtungsregister A
50024		T1 Low Order Latches/Counter (T1C-L)
50025		T1 High Order Counter (T1C-H)
50026		T1 Low Order Latches (T1L-L)
50027		T1 High Order Latches (T1L-H)
50028		T2 Low Order Latches (T2C-L)
50029		T2 High Order Counter (T2C-H)
5002A		Schieberegister (SR)
5002B		Auxiliary Control Register (ACR)
5002C		Peripheral Control Register (PCR)
5002D		Interrupt Flag Register (IFR)
5002E		Interrupt Eingabe Register (IER)
5002F		Output/Input Register B
50030 5FFFF	ACIA u. VIA gespiegelt	
60000 6FFFF	74374	
70000 7FFFF	74374, Start Flip Flop	
80000 FFFFF	frei	

Tabelle 7: Stecker P

Pin	Signal	Pin	Signal
1	VCC	2	VCC
3		4	
5	GND	6	GND
7	DTACK*	8	
9	BERR*	10	CSSYN*
11	VECTIRQ*	12	CLOCK
13		14	
15		16	IRQAKN*
17		18	VPA*
19	AS*	20	CS1*
21		22	POWER-FAIL*
23		24	CS2*
25		26	CS3*
27	RESET*	28	
29		30	HALT*
31	A18	32	A19
33	IPL1	34	IPL0/2
35	A16	36	A17
37	A14	38	A15
39	A12	40	A13
41	A10	42	A11
43	A8	44	A9
45	A6	46	A7
47	A4	48	A5
49	A2	50	A3
51	A0	52	A1
53	D6	54	D7
55	D4	56	D5
57	D2	58	D3
59	D0	60	D1
61	R/W*	62	
63	GND	64	GND

den. Da IC14 und das RAM über eine separate Stromzuleitung verfügen, können diese Bausteine „notversorgt“ werden. Der Aufbau der Power-Fail-Logik und des Akkus erfolgt am besten auf dem Lochrasterfeld.

Anwendungen des EMUF, bei welchen der Anschluß von problembezogener Hardware über VIA, ACIA und die beiden 8-Bit-Ausgangslatches möglich ist, werden am besten über einen 64poligen Stecker (*Tabelle 2*) an die VG-Leiste angeschlossen, dies ermöglicht einen sauberen Aufbau. Als erste Anwendung wird ein „elektronischer Wetterfrosch“ vorgestellt werden. Diese Schaltung läßt sich einfach an den EMUF über die VG-Leiste anstecken. Sie besteht aus einer mehrstelligen Siebensegment-Anzeige, die direkt (mit Widerständen zur Strombegrenzung) an die beiden Latches 74AS374 (IC17, IC18) angeschlossen wird und einem 10-Bit-AD-Wandler mit Anschluß für Temperatur-, Druck- und Feuchtesensoren.

Ein Monitor für den EMUF: MONI-E8

Der Monitor „MONI08“, ursprünglich für den mc-65816-Computer (mc 3/86) geschrieben, wurde an den EMUF angepaßt und heißt nun „MONI E8“. Mit Hilfe von MONI E8 kann Software für den EMUF ausgetestet und fehlerfrei gemacht werden. Der untenstehende Ausdruck *Tabelle 8* des Help-Kommandos von MONI E8 gibt einen Überblick, welche Möglichkeiten für das Debugging (Entwanzen) von Programmen vorhanden sind.

Besitzer von Computern mit einem Assembler für 68000 (natürlich auch Hochsprachen) können mit dem EMUF sehr komfortabel arbeiten. Der EMUF wird über ein V.24-Kabel mit dem Host Computer verbunden, die Programme können auf dem Host editiert, übersetzt und anschließend als Motorola-S-Record zum EMUF übertragen werden. Mit Hilfe von MONI E8 lassen sich eventuell vorhandene Fehler aufspüren. Der Vorteil einer solchen Arbeitsweise liegt darin, daß erst dann, wenn die Software zufriedenstellend läuft, ein EPROM „gebrannt“ wird, und somit die Entwicklungszeit von Software für den EMUF drastisch verkürzt wird.

Tabelle 8: Die Monitor-Kommandos

Command	Key	Parameters
Dump memory	: D	start (end)
Fill memory	: F	start end byte
Compare memory	: C	address1 address2 count
Move memory	: MO	address1 address2 count
Search byte	: SE	start end byte
Load s-records from 1	: L	(string)
Register display	: R	
PC display & change	: P	(value)
SR display & change	: SR	(value)
Dx display & change	: Dx	(value)
Ax display & change	: Ax	(value)
Bx display & change	: Bx	(value)
Breakpoints display	: B	
Clear all breakpoints	: CB	
Step one instruction	: S	(count)
Trace multiple step	: T	(count)
Goto	: G	(address)
Memory Test	: MT	start end
Add/Sub two hex numbers	: A	number1 number2
Calculator	: CA	string

Frank Majewski

Der EMUF86

Nach dem großen Erfolg unserer bisherigen EMUFs, ist es an der Zeit, ein Familienmitglied vorzustellen, das den PC-Benutzern entgegenkommt. Hier ist es: ein EMUF mit 8086-CPU!

EMUF steht für Einplatinencomputer mit universeller Festprogrammierung. Nun wird bei Lesern, die sich schon länger mit Computern beschäftigen, der Begriff 'Einplatinencomputer' am Ende noch immer mulmige Gefühle auslösen: Zu tief sitzt vielleicht ihre Erinnerung an jene ersten Rechnerchen, bei denen jedes Byte des fast unbezahlbaren Speichers über eine kleine Tastatur und Siebensegmentanzeigen 'per Handschlag' persönlich begrüßt werden wollte. Für diese Oldtimer und die überragende Zahl der PC-Besitzer brechen erfreuliche Zeiten herein: Ein professioneller EMUF auf Basis des populären 16-Bit-Prozessors 8086 mit voller Unterstützung durch Interrupt-Controller, Zähler, bis zu 256 KByte EPROM und 128 KByte RAM auf einer ausbaufähigen Doppeleuropakarte!

Um noch kurz bei jenen 'alten Zeiten' zu bleiben: Als die freudig erregte Gemeinde der Fachleute 1981 die Gehäuse der ersten IBM-PCs öffnete, wird es doch so manches verdutzte Gesicht gegeben haben. Denn was war dort statt der vom Großrechner-Giganten erwarteten High-Tech-Granate verschraubt: ein EMUF! Und dann noch einer von der armseligen Sorte: ohne serielle oder parallele Schnittstellen, statt dessen lieber mit Kassettenrecorderanschluß, großzügigen 16 KByte festinstalliertem RAM und 8-Bit-Datenbus.

Im Gegensatz dazu braucht der neue EMUF86 nicht über ein außen angebrachtes Namensschild zu überzeugen: Er ist von Beginn an komfortabel ausgestattet und, sollte es für einen speziellen Fall dennoch einmal nicht ausreichen, leicht über einen flexiblen 16-Bit-Bus erweiterbar (*Bild 1*). Trotz des leistungsfähigen Konzepts bleibt die Hardware aufgrund der Verwendung von Standardbauteilen selbst für Großserien extrem preisgünstig! Wer schon einen Blick auf die Platine des IBM-PC bzw. seiner zahlreichen Zwillinge oder in das dazugehörige technische Handbuch riskiert hat, wird bei der Ausstattung des EMUF86 viele 'alte Bekannte' antreffen. Umgekehrt ist das Experimentieren mit dem EMUF86 auch für bislang unerfahrene PC-Besitzer sehr lohnend, da sich die gewonnenen Erfahrungen sofort auf den PC übertragen lassen. So wäre das sorglose Experimentieren mit Bausteinen im PC nicht ohne Gefahr, da diese immer zumindestens teilweise für z. B. die Steuerung von Floppy und Harddisk oder den Refresh der dynamischen RAMs programmiert sind. Beim EMUF86, der größtenteils die gleichen Bausteine verwendet, ist das vollkommen unkritisch: Der schlimmste aller Fälle wäre ein Programmabsturz, der sich durch Drücken der Reset-Taste problemlos aus der Welt schaffen läßt. Ein darüber hinausgehender Datenverlust ist ausgeschlossen!

Der Verweis auf den PC kommt aber noch aus einem anderen Grund nicht von ungefähr: Er ist mit seiner Flut an Assemblern, Debuggern, Compilern und anderen modernen Software-Tools ein ausgezeichnetes Entwicklungssystem. Der Hardware-Rahmen ist ja, wenngleich auch sehr flexibel, durch die Platine abgesteckt, aber die für jeden Einsatz zu entwickelnde Software spielt die entscheidende Rolle. Das ist schließlich auch die eigentliche Idee hinter dem EMUF-Konzept: Festverschaltete und damit unflexible TTL-'Wüsten' durch leicht anpaßbare Software zu ersetzen.

Software entwickeln mit Komfort

Beim Thema Software werden auch schon wieder einige Bedenken anmelden: „Leicht anpaßbare Programme? Assembler wird er damit wohl doch nicht meinen!" Daß dieser Einwand berechtigt ist, kann jeder ermessen, der sich bereits mit Entwicklung und Pflege eines umfangreicheren Assemblerprogramms beschäftigt hat. Doch gerade auch hier bricht eine neue Ära heran! Die Leistungsfähigkeit moderner 16-Bit-Prozessoren ist so enorm, daß man ohne weiteres etwas davon 'verschenken' darf, um dafür die gestellte Aufgabe soweit wie möglich in einer Hochsprache lösen zu können. Das sind Zeiten!

Das Konzept: Entwickeln des Steuerprogramms auf dem PC mit Screen-Editor und Hochsprachen-Compiler, Übersetzen und dann Überspielen des erzeugten Maschinencodes in das üppige RAM des EMUF, Austesten und im Fehlerfall wieder zurück in den Editor. Der EPROM-Programmierer kommt erst ganz zum Schluß ins Spiel, das Löschgerät bleibt außen vor!

Bei der Auswahl der richtigen Programmiersprache gibt es mehrere wichtige Punkte: Der erzeugte Maschinencode soll kurz und schnell, das Programm leicht auf ein anderes Zielsystem (EMUF, Rechner) übertragbar und ein passender Compiler greifbar sein. Letztendlich will man die gewählte Sprache auch beherrschen (oder schnell erlernen

Tabelle 1: Belegung der VG-Leiste

Pin-Nr.	a	c
1	+ 5 V	+ 5 V
2	D 15	D 14
3	D 13	D 12
4	D 11	D 10
5	D 9	D 8
6	D 7	D 6
7	D 5	D 4
8	D 3	D 2
9	D 1	D 0
10	PCLK	$\overline{\text{RESET}}$
11	$\overline{\text{WRL}}$	Out 1
12	A 19	A 18
13	A 17	A 16
14	A 15	A 14
15	A 13	A 12
16	M/$\overline{\text{IO}}$	Out 2
17	A 11	A 10
18	A 9	A 8
19	A 7	A 6
20	A 5	A 4
21	A 3	A 2
22	A 1	A 0
23	Clk 1	NMI
24	$\overline{\text{IR 6}}$	$\overline{\text{IR 7}}$
25	Out 0	$\overline{\text{IR 5}}$
26	$\overline{\text{RD}}$	$\overline{\text{WRH}}$
27	$\overline{\text{INTA}}$	$\overline{\text{WR}}$
28	Clk 0	Clk 2
29	Clk	$\overline{\text{BHE}}$
30	Gate 0	INTR
31	Gate 2	Gate 1
32	GND	GND

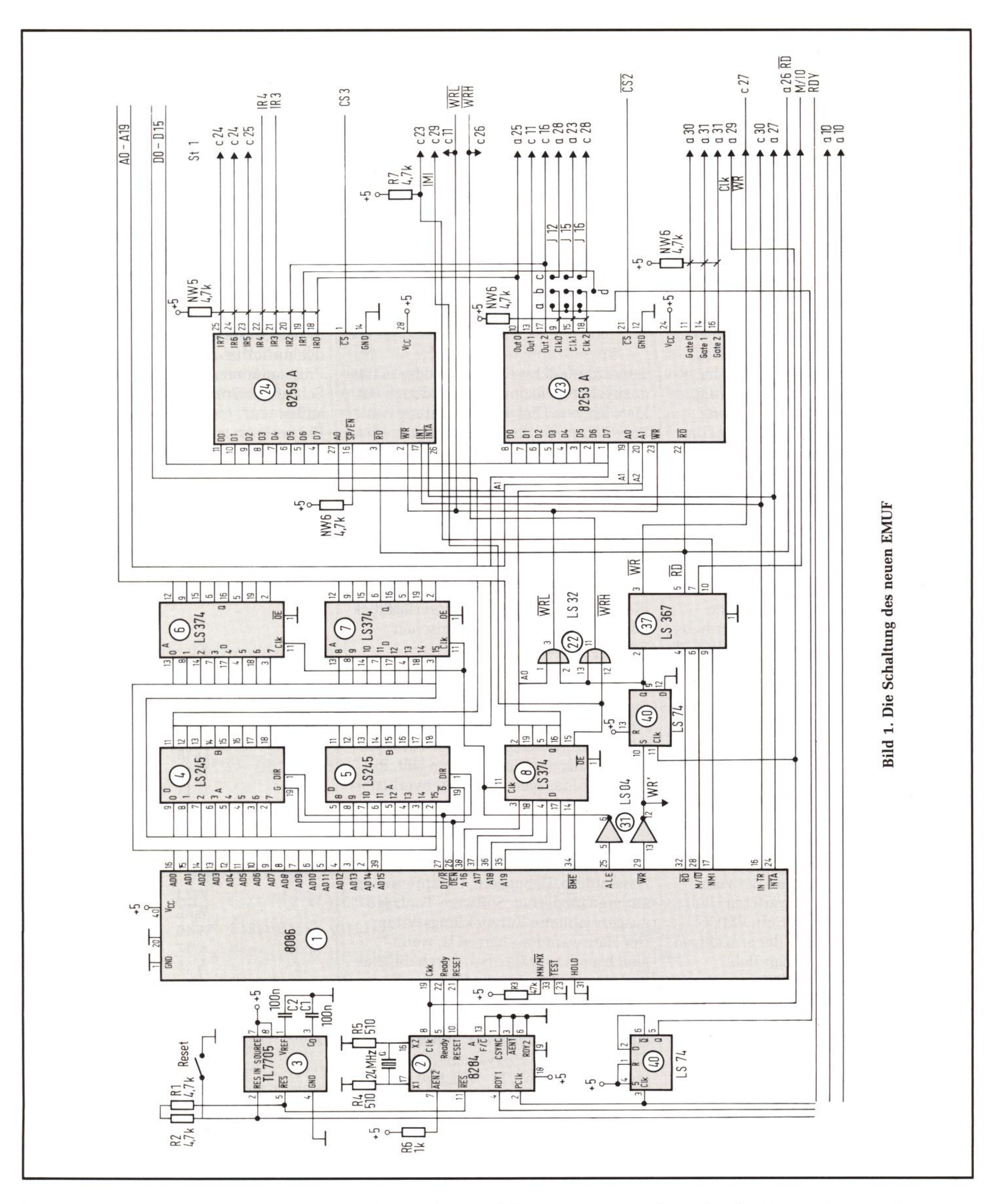


Bild 1. Die Schaltung des neuen EMUF

können). Zur Diskussion stehen u. a. Forth, Pearl, Pascal und „C", Lisp scheidet wohl aus.
In der Regel werden Forth und Pearl aufgrund mangelnder Verfügbarkeit (auf Diskette und im Kopf) keine Verwendung finden. Für Pascal spricht die große Verbreitung von Turbo-Pascal, das in seiner Grundform jedoch leider weder (EP)ROM-fähig noch voll reentrant (wichtig bei Verwendung von Interrupts) ist. 'Kochrezepte' (Patches), die diese Einschränkungen zumindestens teilweise wieder ausbügeln, wurden verschiedentlich veröffentlicht [1]. Übrig

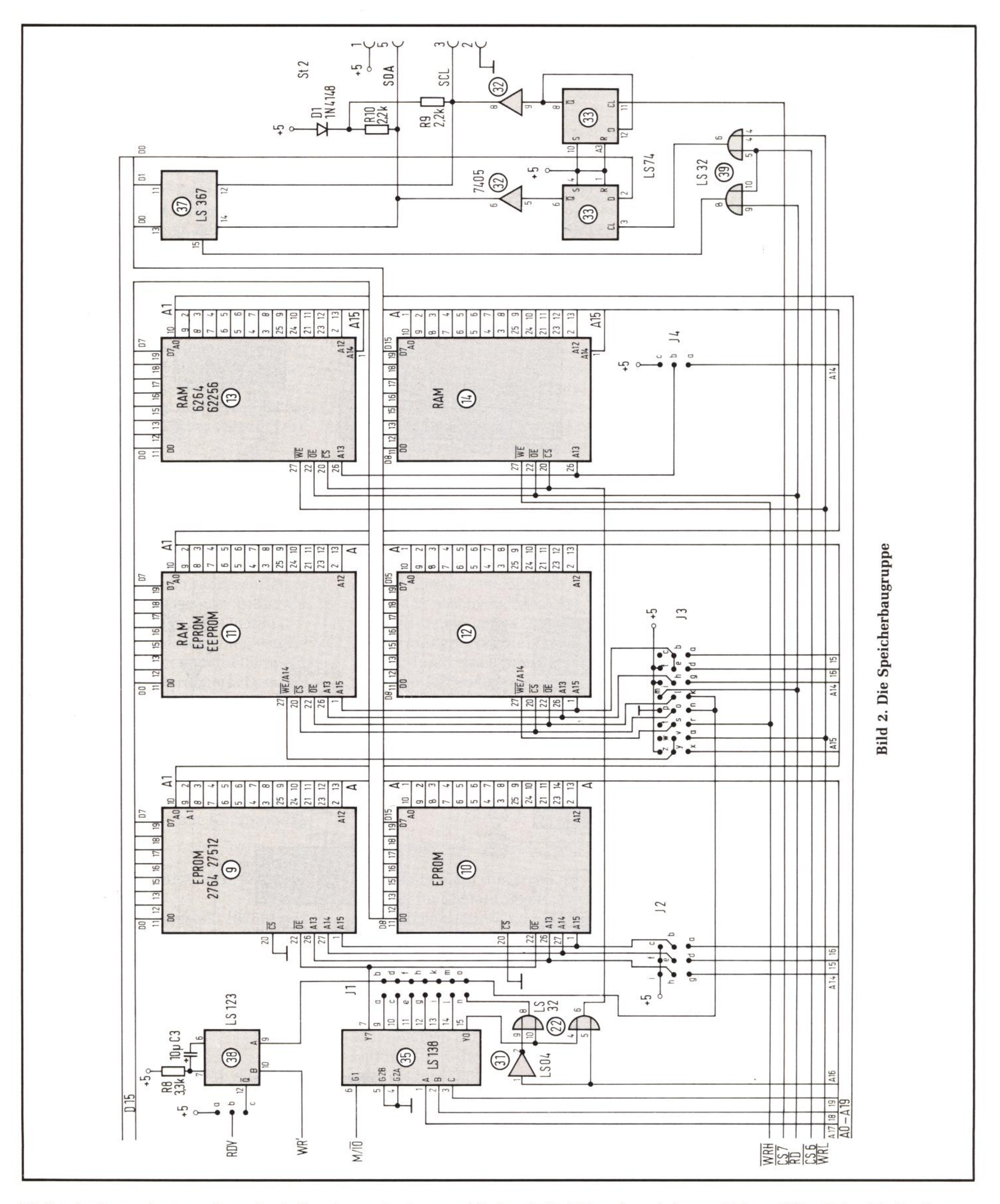


Bild 2. Die Speicherbaugruppe

bleibt, der Leser ahnt es schon, das haßgeliebte „C“! Und in der Tat ist spätestens die Programmierung eines EMUFs der rechte Anlaß, auf diese Hochsprache zu zugehen. Sie verdient es, und der Anwender hat anschließend alle Trümpfe in der Hand.

Ein wichtiger Aspekt ist die große Zahl sehr guter C-Compiler – nicht zuletzt das professionelle Microsoft-C und das vor kurzer Zeit veröffentlichte Turbo-C von Borland.

Pfui Assembler? Aber wer wird denn gleich! Schließlich wird, wer auch das Allerletzte an Leistung aus seinem Pro-

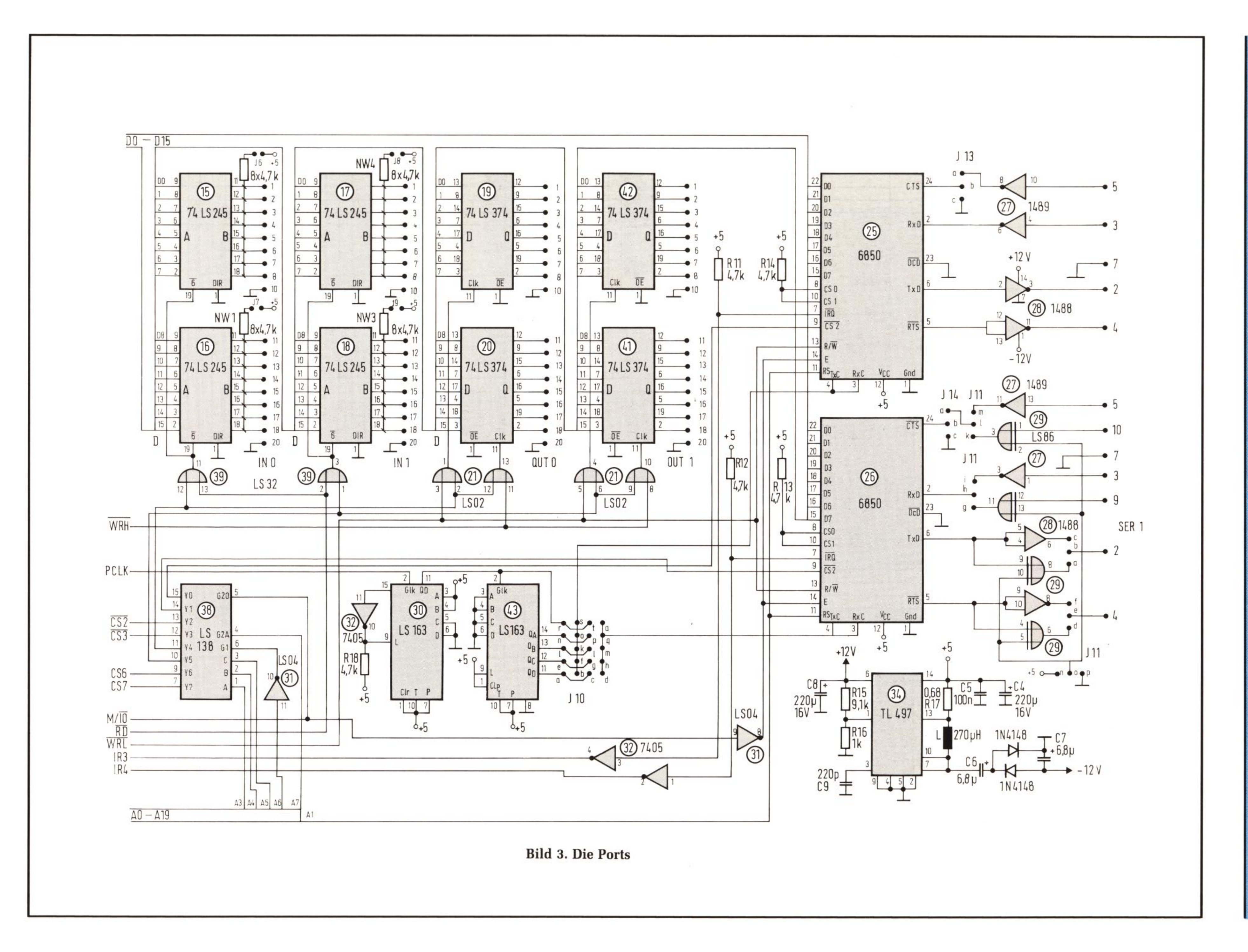


Bild 3. Die Ports

zessor kitzeln will (oder besser muß), intensiven Gebrauch von Interrupts machen und bei der Programmierung z. B. der Interrupt-Service-Routinen auf Assembler zurückgreifen. Zumal die Assemblerprogrammierung des 8086 auf einem System mit nicht mehr als 64 KByte Speicher nur angenehmer als auf einem Z80, 8085 oder gar 6502 ist (und das sage ich als langjähriger Apple-Fan!). Dies liegt daran, daß in einem solchen Fall alle Daten und der Stack noch in einem gemeinsamen Segment (64-KByte-Block) liegen. Etwas unangenehmer wird es, wenn alle vier Segmentregister verschiedene 64-KByte-Blöcke adressieren. Unzumutbar wäre jedoch selbst dieser Fall nicht, der bei einem EMUF aber sowieso die absolute Ausnahme bleiben wird und dem Einsteiger durch einige gute Bücher versüßt wird [2, 3].

Ein „tragfähiger" Kompromiß

Die Wahrheit liegt wie oft in der Mitte, und ein Optimum an effizienter Software kann entstehen, wenn Asssembler und C geschickt kombiniert werden, wobei man die Aushilfskrücke 'Inline-Code', d. h. byteweise in den Quelltext eingestreuten Maschinencode, vergessen darf: Kombinieren meint dann wirklich Linken! Für interessierte Leser sei an dieser Stelle unbedingt auf die Anregungen in [4] verwiesen.

Kompromißlose Hardware

Die Hardware des EMUF86 kommt hingegen vollkommen ohne Kompromisse aus:

- 8086(V30)-CPU mit 8 MHz Taktfrequenz
- Interrupt-Steuerbaustein (8259) mit 8 Eingängen
- Drei 16-Bit-Zähler (8253)
- Zwei serielle Schnittstellen (6850) bis 19200 Bd
- 32 TTL-Eingänge
- 32 TTL-Ausgänge
- I²C-Bus-Adapter
- 6 Steckplätze für EPROM, EEPROM und RAM
- Spannungsüberwachung mit Reset
- 64polige Busleiste
- Nur 5-V-Versorgung erforderlich

Die Auswahl der Komponenten macht deutlich, daß der EMUF86 konsequent für abgeschlossene Automatisierungsaufgaben entwickelt wurde. Dementsprechend ist der Aufbau in moderner CMOS-Technik empfehlenswert. Für CMOS sprechen die niedrige Leistungsaufnahme und damit geringe Aufheizung der gesamten Baugruppe sowie eine gegenüber NMOS deutlich größere Störsicherheit bei dennoch erhöhter Taktfrequenz. Unabhängig davon, ob man den gesamten EMUF in CMOS realisiert, ist es in jedem Fall besser, sich bei der CPU für den V30 von NEC zu entscheiden! Dieser moderne Prozessor ist pin- und befehlskompatibel zum 8086, führt aber Programme bei gleicher Taktfreqenz um ca. 15...20 % schneller aus. Bei Verwendung des erheblich erweiterten Befehlssatzes läßt sich dieser Wert um ein Vielfaches steigern.
Die Arbeit mit diesen Befehlen wird durch Verwendung eines Makro-Assemblers wesentlich erleichtert und ist auf einem EMUF im Gegensatz zum PC sehr sinnvoll, da solche Progamme nicht auf allen 'Kompatiblen' laufen müssen, sondern für eine bestimmte Zielmaschine geschrieben werden. Trotz seiner größeren Leistung ist der V30 sogar noch sparsamer als sein Vorbild: So 'verkocht' der 8086 noch rund 1,7 W elektrische Leistung, wo der V30 mit nur 0,5 W auskommt. Dieser Wert läßt sich noch eimal auf 0,05 W absenken, wenn der V30 über den Halt-Befehl in den Standby-Modus geschickt wird. Aus diesem 'Tiefschlaf' wird er durch einen Interrupt (RESET, NMI oder INT) wieder geweckt. Diese in einem EMUF sehr sinnvolle Einrichtung ist aber nur eine von vielen Verbesserungen gegenüber dem 8086. Eine genauere Beschreibung dieses Prozessors sollte man einem guten Datenbuch entnehmen [5].

Interrupts

Zwei weitere, für die Arbeit mit einem Rechner für Steuerungsaufgaben sehr wichtige Bausteine sind die Chips 8259 und 8253. Der Interrupt-Controller 8259 liegt zwischen dem Prozessor und dessen Peripherie. Erwartet z. B. der 6850 die 'Zuwendung' des Prozessors, weil er ein Zeichen über die serielle Schnittstelle empfangen hat, so legt er seine INT-Leitung auf H-Pegel. Der 8259 ermittelt nun, ob der 6850 eine ausreichend hohe Priorität besitzt, die CPU zu diesem Zeitpunkt zu unterbrechen. Die Priorität eines Bausteins ist dadurch festgelegt, an welche der acht möglichen Interrupt-Eingänge (IR0...IR7) er angeschlossen wurde. Die IR0-Leitung hat die höchste, IR7 die geringste 'Wichtigkeit'.
Läßt der 8259 eine Interrupt-Anforderung zu, so zieht er seinerseits die INT-Leitung der CPU auf H-Pegel. Ist diese zur Interrupt-Behandlung bereit, abhängig davon, ob der Programmierer das I-Flag im Statusregister gesetzt (Unterbrechungen zugelassen) oder gelöscht hat, so bestätigt sie die Anforderung des 8259. Daraufhin legt der Interrupt-Controller ein Byte (Vektor) auf den Datenbus, aus dem der Prozessor die Adresse des Programmteils berechnet, das im genannten Beispiel das empfangene Zeichen aus dem 6850 liest und auf dem Bildschirm darstellt oder in einem Puffer ablegt.

IR0	Zähler 0	(8253A, IC 23)
IR1	Zähler 1	(8253A, IC 23)
IR2	Zähler 2	(8253 A, IC 23)
IR3	SER0	(6850, IC 25)
IR4	SER1	(6850, IC 26)
IR5	frei	(VG-Leiste, Pin 25c)
IR6	frei	(VG-Leiste, Pin 24a)
IR7	frei	(VG-Leiste, Pin 24c)

Bild 4. Interrupt-Quellen

Das Zusammenspiel von CPU und 8259 sieht auf den ersten Blick etwas kompliziert aus, erleichtert aber dem Programmierer die Arbeit ungemein. Er kann seine INT-Service-Routine wie ein normales Unterprogramm schreiben, da sie die Quelle der aktuellen Unterbrechung nicht erst zu ermitteln braucht. Beim 8259 im EMUF86 können die drei Zähler und die beiden 6850 Interrupts erzeugen. Die Leitungen IR5...IR7 stehen an der Erweiterungsleiste zur Verfügung (*Tabelle 1, Bild 4*).

Zähler

Der Zähler/Zeitgeber 8253 enthält drei voneinander unabhängige 16-Bit-Rückwärtszähler, von denen jeder einen Eingang, ein sogenanntes Tor (Gate) und einen Ausgang besitzt. Grundsätzlich generiert also jeder Kanal aus dem Signal an seinem Eingang, geteilt durch eine beliebige 16-Bit-Zahl, ein Ausgangssignal programmierbarer Frequenz.
Über den Gate-Anschluß läßt sich der Zähler triggern, d. h. zu einem festgelegten Zeitpunkt (oder Ereignis) starten. Jeder Kanal kann in sechs möglichen Betriebsarten arbeiten:

M0 Erzeugung periodischer Interrupts nach Eingang einer bestimmten Anzahl von Impulsen.
M1 Programmierbares Monoflop (retriggerbar)
M2 Taktgenerator
M3 Rechteckgenerator

J1: Adreßbereich für IC 11 + IC 12

n–o	10000h–1FFFFh
l–m	20000h–3FFFFh
i–k	40000h–5FFFFh
g–h	60000h–7FFFFh
e–f	80000h–9FFFFh
c–d	A0000h–BFFFFh
a–b	C0000h–DFFFFh

Adreßbereich von IC 13 + IC 14
0–0FFFFh

Adreßbereich von IC 9 + IC 10
E0000h–FFFFFh

J2: EPROM-Typ
2764: b–c, e–f, h–i
27128: b–c, e–f, g–h
27256: b–c, d–e, g–h
27512: a–b, d–e, g–h

J3: Speichertyp für IC 11 + IC 12
6264: e–f, h–i, l–m, n–o, r–s, u–v
65256: (32-KByte-RAM): a–b, g–h, l–m, n–o, r–s, u–v
2764: e–f, k–l, o–p, s–v, y–z
27128: e–f, g–h, k–l, o–p, s–v, x–z
27256: d–e, g–h, k–l, o–p, s–v, x–y
27512: d–e, g–h, k–l, o–p, s–v, x–y

J4: RAM-Typen für IC 13 + IC 14
6264: b–c
65256: a–b

J5:
a–b: wenn keine EEPROMs für IC 11 + IC 12 verwendet werden
b–c: für EEPROMs HN 58064 (J3 muß für 6264 gesteckt sein)

J6, J7, J8, J9
wenn gesteckt, Pull-up-Widerstände an INx aktiv

J10: Baudrate SER0 +SER1

	19200	9600	4800	2400	1200
SER0	r–s	n–o	i–k	e–f	a–b
SER1	t–u	p–q	l–m	g–h	c–d

bei Initialisierung des 6850 mit Clock/16

J11: V.24/TTL-Pegel an SER1

	T×D	$\overline{\text{RTS}}$	R×D	$\overline{\text{CTS}}$
TTL	a–b	d–e	g–h	k–l
V.24	b–c	e–f	h–i	l–m

n–o: TTL-Signale an SER1 invertiert
o–p: TTL-Signale an SER1 nicht invertiert

J12:
a–b: Clk0 = 2 MHz
b–c: Clk0 von ext. über a28 der VG-Leiste

J14:
a–b: $\overline{\text{CTS}}$ von SER1
b–c: $\overline{\text{CTS}}$ mit Masse verbunden

J15:
a–b: Clk1 = 2 MHz
b–c: Clk1 von ext. über a23 der VG-Leiste

J16:
a–b: Clk2 = 2 MHz
b–c: Clk2 von ext. über c28 der VG-Leiste
b–d: Clk2 verbunden mit OUT1 (Zähler 1 und 2 sind hintereinandergeschaltet)

J18:
a–b: $\overline{\text{CTS}}$ von SER0
b–c: $\overline{\text{CTS}}$ mit Masse verbunden

Bild 5. Jumper auf der Platine und ihre Bedeutung

M4 Softwaremäßig triggerbarer Ausgang (bei Nulldurchgang des Zählers)
M5 hardwaremäßig triggerbarer Ausgang (bei Flanke am Gate)
Die Eingänge und Tore sind entweder über den Bus zugänglich, oder es wird als Zähltakt ein auf der Platine vom CPU-Takt abgeleitetes 2-MHz-Signal verwendet. Zusätzlich kann der Ausgang von Zähler 1 auf den zweiten Zähler geschaltet werden, um so einen 32-Bit-Zähler zu erhalten.
Eine Entscheidung für jeden der drei Kanäle trifft man mit den Steckbrücken (Jumper) J12, J15 und J16 (*Bild* 5). Eine detaillierte Beschreibung der sechs Modi dieses sehr leistungsfähigen Bausteins sollte auf jeden Fall den Datenblättern der Hersteller (u. a. NEC, Intel und Siemens) entnommen werden.

Serielle Verbindung mit anderen Computern

Die Verbindung des EMUF86 mit anderen Computern erfolgt seriell per V.24-Schnittstellen, die mit dem 6850 realisiert sind. Hier werden zum erstenmal nicht PC-typische Bausteine eingesetzt, da der standardmäßig verwendete ACE 8250 rund den 5fachen Preis kostet, ohne wesentlich besser zu sein.
Die Pfostenstecker beider Schnittstellen (SER0, SER1) sind normentsprechend belegt und flachbandkompatibel zu je einer DB25-Buchse (*Bild 6*). Dadurch erfolgt bei Einbau des EMUFs in ein eigenes Gehäuse die Verkabelung wie mit einem Modem. Die für die Pegelwandlung von TTL nach V.24 benötigten ±12 V werden auf der Platine durch einen TL497 erzeugt, wodurch man mit nur einer Betriebsspannung für die gesamte Schaltung auskommt. Bei SER0 liegen die Übertragungspegel fest auf V.24-Norm, während bei SER1 die Möglichkeit besteht, auf die Pegelumwandlung zu verzichten und mit TTL-Signalen zu arbeiten. Dies spart zwei ICs und ermöglicht andererseits eine Kommunikation über die parallele Druckerschnittstelle des Entwicklungssystems, wenn dieses über kein serielles Interface verfügt. Diese Form des Datenaustauschs im TTL-Pegel ist problemlos möglich, solange die Verbindungskabel nur eine Länge von wenigen Metern haben.
Abweichend von der Grundeinstellung nach Bild 5 sollte der Aufbau dann gemäß *Bild 7* und *Bild 8* erfolgen.

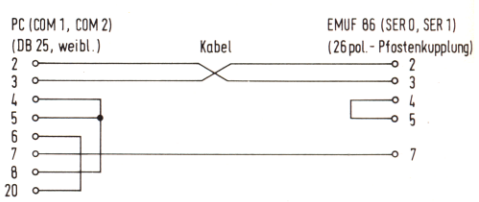
SER 0

Pin	Signal
2	TxD
3	RxD
4	RTS
5	CTS

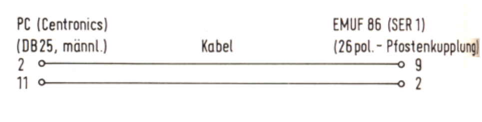
SER 1

Pin	Signal
2	TxD (abh. v. J11b)
3	RxD (V. 24)
4	RTS (abh. v. J11e)
5	CTS (V. 24)
7	Gnd
9	RxD (TTL)
10	CTS (TTL)

Bild 6. Belegung der beiden seriellen Schnittstellen



Kommunikation über Druckerschnittstelle des PC
(nur bei Verwendung eines speziell angepaßten Terminalprogramms möglich!)



Wichtig: Jumper gemäß Bild 6 einstellen!

Bild 7. Der PC kann über die serielle Schnittstelle oder über die Druckerschnittstelle an den EMUF angeschlossen werden

Jumper	Funktion	Stellung
J11	TTL-Pegel an T×D	a–b
J11	TTL-Pegel an $\overline{RTS}$	d–e
J11	TTL-Pegel an R×D	g–h
J11	TTL-Pegel an $\overline{CTS}$	k–l
J11	TTL-Pegel nicht invertiert	o–p
J10	Baudrate 1200 Bd	c–d
J14	Kein Hardware-Handshake	b–c

Bild 8. Jumper-Einstellung bei Kommunikation über die Druckerschnittstelle des PC

Da die Kommunikation in der Regel das Software-orientierte XON-/XOFF-Protokoll verwendet, besteht die Möglichkeit, über J13 (SER0) und J14 (SER1) das $\overline{CTS}$-Handshake-Signal dauerhaft zu aktivieren (L-Pegel). Es ist zu beachten, daß die TTL-Signale bei SER1 unabhängig von den mit Jumper J11 gewählten Pegeln mit auf dem Pfostenstecker (Belegung siehe Bild 4) liegen. Dadurch besteht auch die Möglichkeit, externe Umsetzer auf z. B. RS485, LWL oder 20-mA-Stromschleife anzuschließen. Dazu kann es eventuell nötig werden, die TTL-Signale vorher mittels J11, Pin o zu invertieren.
Ähnlich flexibel läßt sich die Baudrate einstellen. Sie erfolgt für jeden der beiden Kanäle getrennt durch J10 (Bild 5). Zusammen mit dem internen Teiler des 6850 ergeben sich so sieben verschiedene Übertragungsgeschwindigkeiten zwischen 300 und 19 200 Bd. In Bild 7 wird detailliert gezeigt, wie man PC und EMUF verbinden kann.

„Altmodische" Ein-/Ausgabe

Die Ein-/Ausgabe-Ports sind mit einfachen TTL-Bausteinen realisiert, was auf den ersten Blick etwas altmodisch aussieht, jedoch flexibel und preiswert die Möglichkeit bietet, bei Bedarf 16 Bit breite Daten in einem Zyklus zu lesen (IN0, IN1) oder ausgeben (OUT0, OUT1). Dazu werden jeweils zwei 8-Bit-Port-Bausteine zusammengefaßt und über einen 20poligen Pfostenstecker (*Bild 9*) herausgeführt, an die mit Flachbandkabel unter anderem passende E/A-Module für die 24-V-Steuerungstechnik angeschlossen werden können. Mehrere solcher Erweiterungsmodule sind bereits realisiert.
Die Adressen, unter denen die Ports angesprochen werden können, zeigt *Bild 10*. Die Eingänge sind wahlweise mit Pull-up-Widerständen bestückbar, können aber jederzeit in 8-Bit-Gruppen wieder davon freigeschaltet werden (J6...J9). Die Stiftleisten IN0 und OUT0 sowie IN1 und OUT1 sind so angeordnet, daß durch Bestücken des Flachbandkabels mit zwei Pfostenkupplungen die Datenausgänge zurückgelesen werden können.

	D1	D3	D5	D7	NC	D9	D11	D13	D15	NC
Pin 2	o	o	o	o	o	o	o	o	o	o
Pin 1	o	o	o	o	o	o	o	o	o	o
	D0	D2	D4	D6	NC	D8	D10	D12	D14	NC

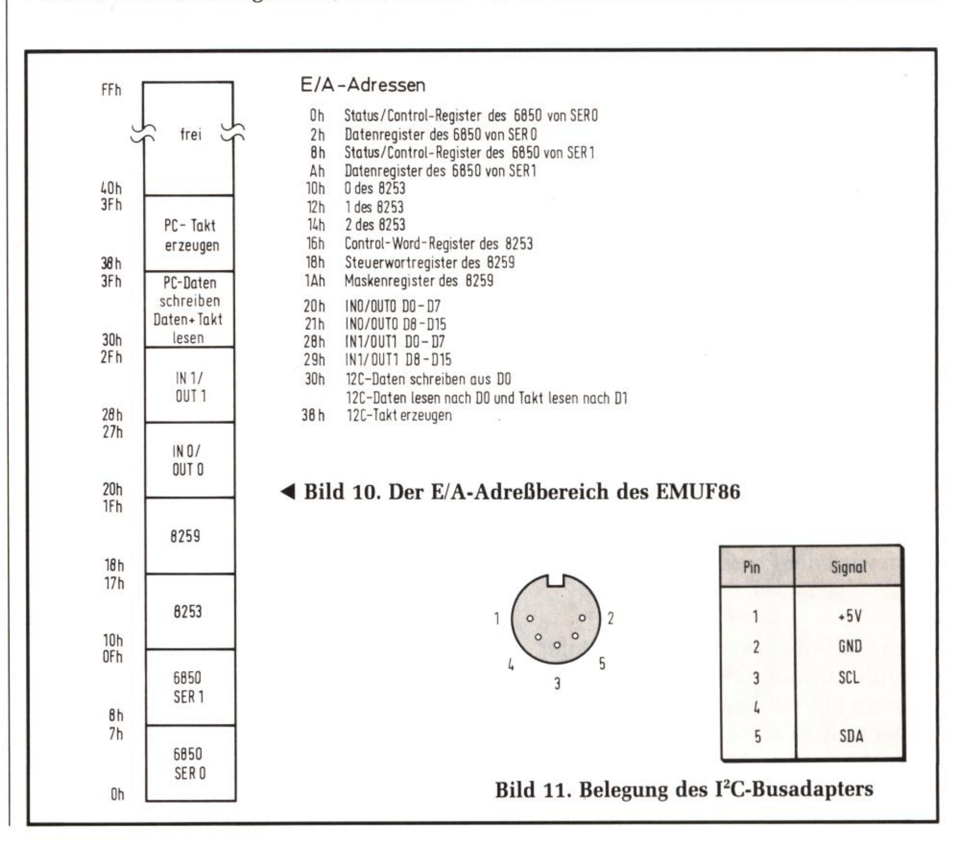
Bild 9. Belegung der E/A-Stiftleisten

Mit I²C-Bus!

I²C! Nein, diesmal nicht das Neueste aus der Elektrotechnik, sondern der Name eines bidirektionalen 2-Bit-Bussystems. Dieser 'Bus' ist auf eine 5polige DIN-buchse (*Bild 11*) herausgeführt und ermöglicht den Anschluß weiterer Baugruppen. Nach I²C-Konvention werden die angeschlossenen Systeme untereinander durch eine Daten- und Taktleitung verbunden. Die Kommunikation erfolgt dann bitseriell, ist aber nicht zeitsynchronisiert (wie z.B. V.24), sondern der Sender synchronisiert seine Daten selbst, indem er fortlaufend nach dem Herausschreiben eines Bits die Taktleitung invertiert.
Zur Realisierung des I²C-Anschlusses ist ein 2-Bit-Port vorgesehen. Wird dieser Port auf Adresse 30h gelesen, so befindet sich in D0 der momentane Wert von Pin 5 (SDA) der DIN-Buchse und in D1 der Wert an Pin 3 (SCL). Das Bit in D0 ist ein gültiges Datenbit, wenn die Taktleitung L-Pegel führt und sich ihr Zustand seit dem letzten Zugriff geändert hat. Umgekehrt schreibt der Sender das auszugebende Bit D0 in die Adresse 30h und invertiert anschließend durch bloßen Zugriff (lesend oder schreibend) auf die Adresse 38h die Taktleitung. Damit signalisiert er dem Empfänger die Gültigkeit des geschriebenen Bits. Ein Datenaustausch ist jedoch nicht ausschließlich mit Baugruppen möglich, die speziell für den I²C-Bus ausgelegt sind, sondern auch mit C-Bus-Geräten oder anderen 2-Draht-Bussen. Meist ist dann bei mehreren Stationen ein Hinzufügen von Freigabeleitungen nötig, die über OUT0 und OUT1 leicht erzeugt werden können.

Diverses

Neben den auffälligen Bestandteilen verrichten auch noch etliche andere Bausteine ihren Dienst, die von 'Software-Technikern' unbeachtet bleiben werden, da sie nicht programmierbar sind. Trotzdem läuft ohne sie gar nichts! Da ist z.B. ein TL7705, der die 5-V-Spannungsversorgung überwacht. Sollte sie einen Wert von 4,8 V unterschreiten, wird sicherheitshalber ein Reset ausgelöst, um ein Weiterrechnen mit falschen Daten zu



◀ Bild 10. Der E/A-Adreßbereich des EMUF86

Bild 11. Belegung des I²C-Busadapters

verhindern. Damit dies nicht geschieht, sollte das Netzteil für eine Belastung mit 2 A ausgelegt werden, wovon der EMUF, abhängig vom Umfang der Bestückung rund 1,5 A aufnimmt. Der 'Saft' kann sowohl über die Busleiste als auch per separatem seitlichem Anschluß zugeführt werden.
Während der Entwicklungszeit wird es in den meisten Fällen möglich sein, den EMUF86 über das PC-Netzteil zu versorgen. Dies hängt natürlich von dessen Belastbarkeit und der Zahl der Erweiterungskarten im PC ab. Spätestens nach Fertigstellung des Steuerprogramms braucht der EMUF jedoch eine eigene Stromversorgung, für die sich z. B. nach einer kleinen Modifikation die POW5V-Baugruppe des NDR-Klein-Computers gut eignet. Diese Schaltung hat sich vielfach bewährt und ist mit einem Preis von rund 30 DM auch sehr preiswert.

Großzügiger Speicher

Speichern kann der EMUF86 reichlich! Er ist mit sechs Sockeln ausgestattet, von denen jeweils zwei nur entweder EPROMs (bis27512) oder RAMs (bis 62256) aufnehmen können (*Bild 12*). Das dritte Paar, die beiden mittleren Sockel (IC11 und IC12), sind mit J1 auf sieben verschiedene Startadressen einzustellen, u. a. so, daß je nach Bedarf ein zusammenhängender ROM- oder RAM-Bereich entsteht. Außerdem lassen sich in dieses Sockelpaar auch EEPROMs (elektrisch löschbare PROMs) einsetzen. Hierbei können sowohl die RAM-ähnlichen Data-Polling-Typen eingesetzt werden, als auch die billigen Typen, die das $\overline{WR}$-Signal für 10 ms auf low benötigen. Dazu wird der Prozessor entsprechend lange angehalten.

0h...0FFFFh	RAM
10000h...DFFFFh	RAM, EPROM oder EEPROM (64-KByte-Bereich mit J1 wählen)
E0000h...FFFFFh	EPROM

Bild 12. Speicheradressen und mögliche Bestückung

Da die CPU nach einem Reset mit der Ausführung eines Programms an der Adresse FFFF0h beginnt, müssen mindestens die EPROM-Sockel IC9 und IC10 bestückt werden. Normalerweise wird an dieser Stelle das Monitorprogramm eingesetzt, das die Initialisierung der Peripheriebausteine besorgt und darüber hinaus einige nützliche Befehle zur Verfügung stellt (*Bild 13*). Der Monitor ermöglicht unter anderem das Laden eines Programms ins RAM oder EEPROM, das auf einem anderen Rechner erstellt wurde. Darüber hinaus lassen sich auch Register, Speicher und E/A-Bausteine gezielt 'bearbeiten', wobei der PC zum Terminal degradiert wird. Ein entsprechendes Programm für PC-kompatible Rechner mit folgenden Eigenschaften ist verfügbar:

- Terminal-Emulation mit Kommunikation über V.24.
- Programm-'Download' zum EMUF86.
- Daten-'Upload' vom EMUF86 mit Aufzeichnung auf Floppy oder Harddisk.
- Ausgabe der vom EMUF86 aufgenommenen Daten auf den Drucker.
- Umwandeln einer MS-DOS Maschinencode-Datei in zwei getrennte Dateien für das Programmieren von EPROMs (ODD, EVEN).

Adreßbelegung

Die Adreßdecodierung ist recht einfach gehalten: Der E/A-Bereich wird nur auf 256 Port-Adressen ausdecodiert, wovon auf der Platine nur 64 belegt sind. Die restlichen 192 bleiben frei für Erweiterungen über den Bus. Die Beschränkung auf 256 Adressen ist sinnvoll, da so unnötige Decodierlogik vermieden wird. Im verbleibenden Adreßraum lassen sich noch mehr als zwei Dutzend weiterer Port-Bausteine unterbringen. Programmiertechnisch ergibt sich ebenfalls ein Vorteil, da die ersten 256 Ports direkt adressiert werden können. Ein vorheriges Laden des DX-Registers beim Zugriff auf eine 16-Bit-Port-Adresse ist nicht nötig. Zum Beispiel:

```
OUT 10,al ; Schreibe Akku in den Port
          ; auf Adresse 10
```

gegenüber:

```
MOV DX,300 ; lade DX-Register mit Port
           ; auf adresse 300
OUT DX,AL  ; indirekt adressierter
           ; Port-Zugriff
```

Außer den TTL-Ports werden die E/A-Bausteine nur auf den geraden Adressen erreicht. Das bedeutet, daß z. B. das Statusregister des 6850 auf Adresse 0 liegt, sein Datenregister aber nicht auf 1, sondern auf 2. Eine genaue Aufstellung der Portadressen aller im EMUF86 verwendeten ICs enthält Bild 10.

Enter ASCII-Text Into Memory	: A
Display Memory	: D Start Ende
Set ES-Register	: E Wert
Fill Memory	: F
Execute a User Program	: G
ADD & SUB two Hex Numbers	: H
Read Port	: I
Load Program	: L
Move Memory	: M
Test Memory	: N
Output to a Port	: O
Register Display	: R
Enter Hex Data into Mem.	: S
Compare two Memory Blocks	: V

Bild 13. Liste der Monitorkommandos

Nun wird der EMUF zusammengebaut

Der Zusammenbau des EMUF86 sollte in jedem Fall sehr gewissenhaft erfolgen, da die Fehlersuche auf einer solch umfangreichen Platine kein 'Zuckerlecken' ist. Es wäre z. B. sehr sinnvoll, wenn Anfänger einen Freund mit mehr Erfahrung (und im Fehlerfall mit Oszilloskop) um Mithilfe bäten. Die Verwendung von qualitativ guten Sockeln ist nicht nur bei den 'großen' ICs gute Schule, sondern hat sich allgemein als sehr sinnvoll herausgestellt. An dieser Stelle ein paar Mark sparen zu wollen kann hartnäckigen Ärger einbringen.

Platinen, Bausätze und auch Fertiggeräte können vom Elektronikladen, Detmold, bezogen werden. Die Stückliste zeigt *Bild 14* (siehe Seite 98).

Hinweise für das Bestücken umfangreicher Platinen wurden in mc schon vielfach gegeben, deshalb das Wesentliche in Kürze: Zuerst die wenigen passiven Bauteile (Widerstände, Kondensatoren, Quarz u. ä.), dann Sockel, Stiftreihen und Erweiterungsleiste einlöten. Als letztes kommt die einem Programmierer ewig dubiose Spule für den ±12-V-Schaltregler an die Reihe: Sie besteht aus nichts weiter als einem kleinen Ring aus Ferrit (Durchmesser 12 mm), durch den etwa acht Windungen isolierten Kupferdrahts (bis 1 mm Durchmesser) gezogen werden. Das war's schon!

Bevor das ganze (*Bilder 15* und *16*) zum ersten Mal unter Spannung gesetzt wird, müßen noch die der gewählten Ausbauvariante entsprechenden Jumper gemäß

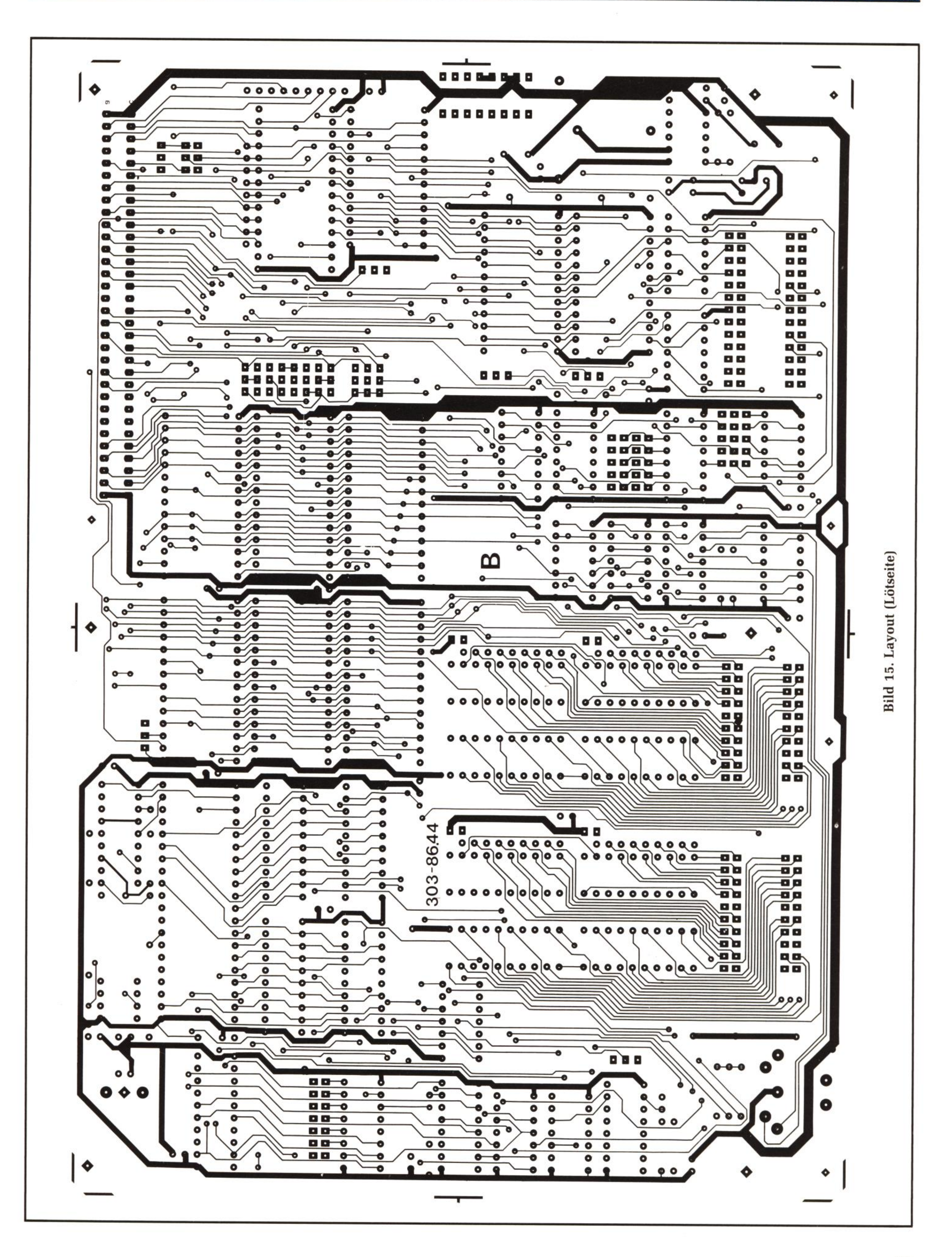


Bild 15. Layout (Lötseite)

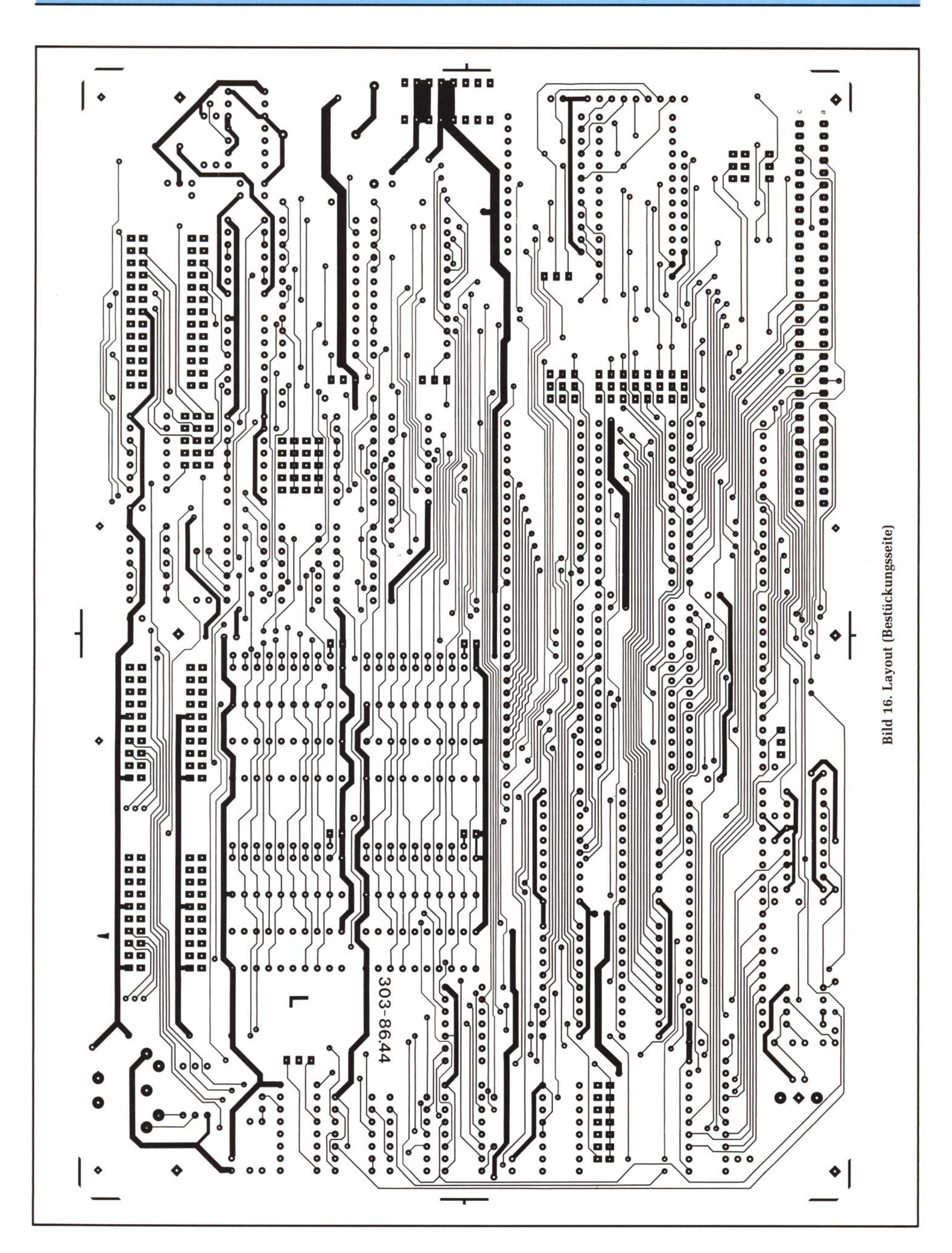


Bild 16. Layout (Bestückungsseite)

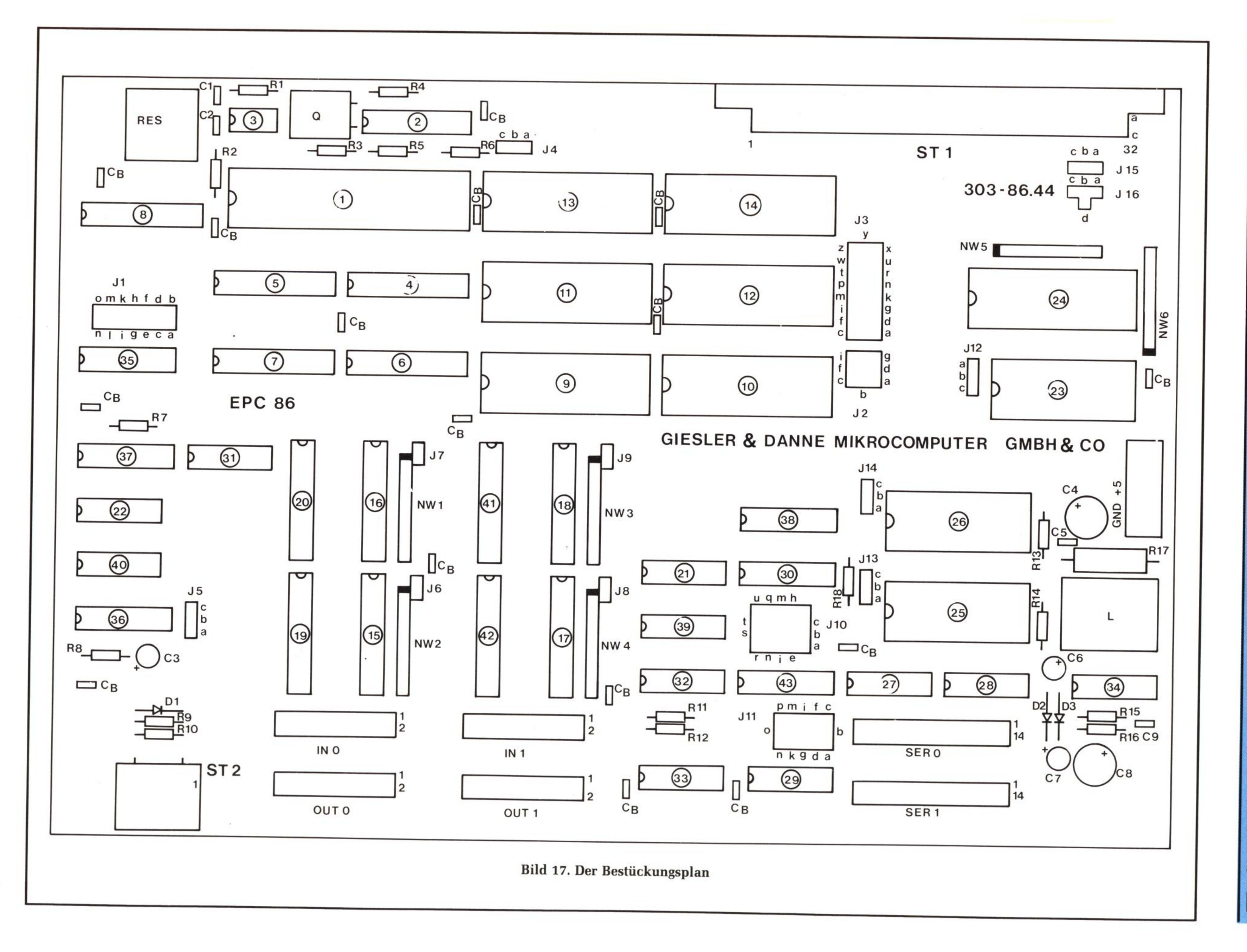


Bild 17. Der Bestückungsplan

Stück	Aufdruck	Beschreibung
1	IC 1	8086-2 o. V30-8
1	IC 2	8284A
1	IC 3	TL7705
6	IC 4, 5, 15, 16, 17, 18	74LS245
7	IC 6, 7, 8, 19, 20, 41, 42	74LS374
6	IC 9, 10, 11, 12, 13, 14	Speicher
1	IC 21	74LS02
2	IC 22, 39	74LS32
1	IC 23	8253
1	IC 24	8259A
2	IC 25, 26	6850
1	IC 27	MC1489
1	IC 28	MC1488
1	IC 29	74LS86
2	IC 30, 43	74LS163
1	IC 31	74LS04
1	IC 32	7405
2	IC 33, 40	74LS74
1	IC 34	TL497
2	IC 35, 38	74LS138
1	IC 36	74LS123
1	IC 37	74LS367
1	Q	Quarz 24 MHz HC-18/U Serienresonanz
19	C1, C2, C5, CB	Kondensator 100 nF RM 2,54 mm
1	C3	Tantal-Elko 10 µF/16 V
1	C4, C8	Elko 220 µF/16 V stehend RM 5,08 mm
2	C6, C7	Tantal-Elko 6,8 µF/16 V
1	C9	Kondensator 200 pF Keramik
9	R1, R2, R3, R7, R11, R12, R13, R14, R18	Widerstand 4,7 kΩ
2	R4, R5	Widerstand 510 Ω
1	R6, R16	Widerstand 1 kΩ
1	R8	Widerstand 3,3 kΩ
2	R9, R10	Widerstand 2,2 kΩ
1	R15	Widerstand 9,1 kΩ
1	R17	Widerstand 0,68 Ω/1 W
5	NW1, NW2, NW3; NW4, NW5	SIL-Array 8× 4,7 kΩ
1	NW6	SIL-Array 4× 4,7 kΩ
3	D1, D2, D3	Diode 1N4148
1	L	Spule 270 µH
1	zu IC 2	IC-Sockel DIL 18
3	zu IC 23, 25, 26	IC-Sockel DIL 24
7	zu IC 9, 10, 11, 12, 13, 14, 24	IC-Sockel DIL 28
1	zu IC 1	IC-Sockel DIL 40
4	IN0, IN1, OUT0, OUT1	Stiftleiste 2× 10pol.
2	SER0, SER1	Stiftleiste 2× 13pol.
1	J1	Stiftleiste 2× 7pol.
1	J2	Stiftleiste 3× 3pol.
1	J3	Stiftleiste 3× 8pol.
7	J4, J5, J12, J13, J14, IR012, Gate012	Stiftleiste 1× 3pol.
4	J6, J7, J8, J9	Stiftleiste 1× 2pol.
1	J10	Stiftleiste 4× 5pol.
1	J11	Stiftleiste 3× 5pol.
1	NMI/IR	Stiftleiste 1× 4pol.
1	–	Buchsenleiste 8pol.
20	–	Jumper
1	RES	Siemens-Tastenelement mit Tastenkappe
1	ST1	VG64-Messerleiste abgew./gerade
1	ST2	Diodenbuchse 5pol. zur Platinenmontage
1	–	Leiterplatte

Bild 14. Die Stückliste

Bild 5 gesteckt und eine sorgfältige Prüfung der Löt- und Bestückungsarbeit durchgeführt werden. Nach dem Anschluß der Versorgungsspannung ist zu kontrollieren, ob an allen Sockeln Masse und +5 V an den richtigen Pins (und sonst keinen!) anliegen. Nach dem anschließenden Trennen von der Betriebsspannung kann man beginnen, die ICs in ihre Sockel zu setzen, wobei peinlichst darauf geachtet werden muß, alle Bausteine richtig herum und auch alle Pins unverbogen zu 'versenken'. Hier beugt ein Blick auf den Bestückungsplan *Bild 17* unnötigen Kopfschmerzen wirksam vor.

Beim Arbeiten mit CMOS-ICs sollten die bekannten Vorsichtsmaßnahmen, wie Transport nur in speziellen Vorrichtungen und vor dem Berühren ein sich 'Entladen' über geerdete Stahlteile, unbedingt eingehalten werden. Nach Fertigstellung und Test sollte der Einbau in ein Blick auf den Bestückungsplan *Bild 17* unnötigen Kopfschmerzen wirksam vor.

Nun wird's spannend

Nach dem Verbinden von EMUF86 und PC wird das Netzteil erneut aktiviert. Befindet sich nun das Monitorprogramm in den EPROMs und läuft ein Terminalprogramm auf der Gegenseite, so erscheint (hoffentlich) folgende Meldung:

EPC 86 – Monitor

Dies signalisiert dem faszinierten Betrachter das erfolgreiche Durchlaufen der Initialisierungsroutine im EMUF86. Sollte diese Bestätigung auch nach wiederholtem Drücken der Reset-Taste ausbleiben, muß sich der Besitzer auf den 'dornigen Weg' der Fehlereingrenzung begeben. Nach einem erfolgreichen Starten des Monitors stehen dann die Befehle aus Bild 13 zur Verfügung.

Literatur

[1] c't 12/85, S.74 und 2/87, S.72

[2] *Sargent,Shoemaker,Stelzer:* Assemblersprache und Hardware des IBM PC. Addison-Wesley.

[3] *Bradley:* Programmieren in Assembler, Hanser-Verlag.

[4] *Biggerstaff:* System Software Tools. Prentice-Hall.

[5] V30 User's Manual. NEC Deutschland.

uerscheinung
s große MS-DOS-Profi-Arbeitsbuch
-DOS detailliert in allen Versionen aufbe-et. Von D. Smode. 400 S., 67 Abb., Tab., tings, Lwstr-geb. DM 68,–
N 3-7723-8681-4
r wird die Schnittstelle Anwenderpro-mm/MS-DOS ausführlich und absolut xisgerecht beschrieben. Die neuesten sionen von MS-DOS sind gleichzeitig die undlage für die bereits angelaufene neue Generation IBM-AT.



Neuerscheinung
Rechner modular
Der NDR-Klein-Computer – selbstgebaut und programmiert. Von R.-D. Klein. 423 S., 410 Abb., 25 Tab., Lwstr-geb. DM 68,–
ISBN 3-7723-8721-7
Nach der Lektüre verfügt der Leser über ein modernes und leistungsfähiges Computer-system. Dieses garantiert durch seinen mo-dularen Aufbau nicht zu veralten und kann leicht erweitert werden.

leues vom ranzis-Computer-uchmarkt

uerscheinung
th 83
e gründliche Einführung in die Forth-Ver-n – auch für PCs. Von R. Zech. 371 S., Abb., Lwstr-geb. DM 78,–
N 3-7723-8621-0
grammierer, die diesen Band durchge-n, sind in der Lage, selbständig mit dem uen Sprach-Dialekt Forth 83 zu arbeiten d sich weitere Systemkenntnisse beliebi-Tiefe anzueignen.

Neuerscheinung
Datenbanksysteme – Auswahl und Einsatz
Wege zur individuellen Datenverwaltung. Von A. Janson. 186 S., 104 Abb., Lwstr-kart. DM 38,–
ISBN 3-7723-8671-7
Wer den Band durchgearbeitet hat, verfügt über ein solides Grundwissen zum Thema Datenbanken und ist in der Lage, ein für seine Zwecke geeignetes System auszuwäh-len und einzusetzen.

uerscheinung
s Betriebssystem Geos
neue C-64-Praxis. Von D. Schoder. S., 52 Abb., Lwstr-kart. DM 38,–
N 3-7723-8631-8
ser Band richtet sich an alle Benutzer des itverbreiteten und erfolgreichen Compu-s C 64. Nach dem Durcharbeiten ist der er in der Lage, Geos-like-Programme zu wickeln.

Neuerscheinung
Schneider CPC: Dateiverwaltung
Eine Software-Sammlung. Von L. Miedel. 183 S., 11 Abb., Lwstr-kart. DM 38,–
ISBN 3-7723-8691-1
Das Verstehen umfangreicher Programme steht im Vordergrund dieser Software-Sammlung. Mit dem vorliegenden Buch hält der Leser eine aktuelle Trickkiste für seinen CPC in der Hand.

uerscheinung
mal für Einsteiger
e leistungsfähige Sprache zwischen sic und Pascal. Von R. Busch. 115 S., Abb., Lwstr-kart. DM 38,–
N 3-7723-8601-6
Autor bringt den Leser von einer einfa-en Kontaktaufnahme mit der Program-ersprache über die Prozeduren und Funk-nen sowie Grafik und Musik bis hin zu satzbereiten Daten-Programmen ans Ziel. bei wird das Comal-Modul für den C 64/ eingesetzt.

Neuerscheinung
Basiswissen der Datenkommunikation
Begriffe – Methoden – Komponenten. Von G. Kafka. 226 S., 136 Abb., 27 Tab., Lwstr-geb. DM 68,–
ISBN 3-7723-8501-X
Jeder, der sich intensiv mit der Kommunika-tionstechnik befaßt, sei es als Entwickler oder als technisch orientierter Anwender, findet hier die aktuelle Zusammenfassung der entscheidenden Fakten.

tell-Coupon für kostenlosen NZIS-Gesamt-Buchkatalog

/Beruf

e

EMUF 2



Franzis-Verlag GmbH
Karlstraße 37–41
8000 München 2
Telefon 5117-1